Par **Kaci BADER**

Tolérance aux fautes pour la perception multi-capteurs : application à la localisation d'un véhicule intelligent

Thèse présentée
pour l'obtention du grade
de Docteur de l'UTC

Soutenue le 05 décembre 2014
Spécialité : Technologie de l'Information et des Systèmes

D2161

Tolérance aux fautes pour la perception multi-capteurs : Application à la localisation d'un véhicule intelligent

Kaci BADER

Thèse soutenue le 5 décembre 2014 devant le jury composé de :

Président:

Isabelle FANTONI
Directrice de recherche au CNRS
Univ de Technologie de Compiègne

Rapporteurs:

Michèle ROMBAUT
Professeur des universités
Univ Joseph Fourier de Grenoble

Frédéric VANDERHAEGEN
Professeur des universités
Univ de Valenciennes

Examinateurs:

Isabelle FANTONI
Directrice de recherche au CNRS
Univ de Technologie de Compiègne

Jérémie GUIOCHET
Maître de conférences
Univ de Toulouse

Directeurs de Thèse:

Walter SCHÖN
Professeur
Univ de Technologie de Compiègne

Benjamin LUSSIER
Enseignant chercheur
Univ de Technologie de Compiègne

Université de Technologie de Compiègne

Laboratoire Heudiasyc UMR CNRS 7253

05 - 12 - 2014

Avant propos

Ces travaux de thèse sont réalisés dans le laboratoire Heudiasyc, au sein de l'équipe ASER. Leur financement est assuré par une allocation ministérielle.

Je tiens à exprimer mes plus sincères remerciements à Monsieur Ali CHARARA directeur de l'unité mixte de recherche CNRS 7253 HeuDiaSyc et Monsieur Philippe BONNIFAIT responsable de l'équipe Automatique, Systèmes Embarqués, Robotique (ASER) qui m'ont accueilli dans des meilleurs conditions pour réaliser cette thèse.

Je remercie Madame Isabelle FANTONI, Directrice de recherche à l'Université de Technologie de Compiègne, pour avoir présidé mon jury de thèse. Mes remerciements vont également à Michèle ROMBAUT Professeur UJF à l'Institut de Technologie de Grenoble, et Frédéric VANDERHAEGEN Professeur à l'Université de Valenciennes pour le temps passé à relire cette thèse en tant que rapporteurs et leurs remarques pertinentes vis à vis de mon travail. Je remercié également Monsieur Jérémie GUIOCHET Maitre de conférence au LAAS qui m'a fait l'honneur d'accepter de participer au jury de ma thèse, pour la grande pertinence de ses questions et remarques.

Un grand merci à Walter SCHÖN et Benjamin LUSSIER mes directeurs de thèse pour la confiance qu'ils m'ont accordé pour réaliser ce travail, pour m'avoir beaucoup appris, pour leurs conseils et leur soutien surtout dans les moments difficiles. Je leur en suis reconnaissant de m'avoir permis de poursuivre mes recherches tout en étant toujours présent pour m'aider.

Je remercie mes collègues du laboratoire Heudiasyc pour la convivialité ainsi que la bonne ambiance qui régnait au laboratoire. Je remercie particulièrement Gérald pour le temps qu'il a su me consacrer et pour son expertise. Je remercié tout mes amis particulièrement Djamel BOUCHENEB, Patricia MIAR, Abderrahim SAHLI pour leur soutien.

Finalement je voudrais remercier plus particulièrement mes parents, mes frères et sœurs pour leur soutien et leur confiance en moi. Le parcours que j'ai fais jusqu'à aujourd'hui n'aurait pas été possible sans leur soutien et leurs conseils.

Table des matières

Liste des figures

Liste des tableaux

Liste des algorithmes

Publications

- **Conférences internationales avec acte et comité de lecture**
  - ◇ **Bader, K.** and Lussier, B. and Schön, W. A Fault Tolerant Architecture For Data Fusion Targeting Hardware And Software Faults , IEEE Pacific Rim International Symposium on Dependable Computing PRDC, Singapour, Novembre 2014.
  - ◇ **Bader, K.** and Lussier, B. and Schön, W.Functional Diversification for Software Fault Tolerance in Data Fusion : a real Application on Kalman Filters for Mobile Robot Yaw Estimation. European Safety and Reliability Conference, ESREL 2015.
- **Conférences nationales avec acte et comité de lecture**
  - ◇ **Bader, K.** and Lussier, B. and Schön, W. Localisation Multi-capteurs Tolérante aux Fautes : Validation expérimentale sur un cas réel, QUALITA, Nancy, France, Mars 2015.
  - ◇ **Bader, K.** and Lussier, B. and Schön, W. Architecture de filtre de Kalman tolérante aux fautes pour la localisation des robots mobiles LAMBDAMU, Dijon, France, Octobre 2014.
  - ◇ **Bader, K.** and Lussier, B. and Schön, W. Tolérance aux Fautes par Fusion de Données : Un Cas d'Étude, QUALITA, Compiègne, France, Mars 2013.
- **En cours de finalisation**
  - ◇ **Bader, K.** and Lussier, B. and Schön, W. Fault Tolerance by data fusion : a case study. (Revue internationale : Information Fusion).

Resumé

La perception est une entrée fondamentale des systèmes robotiques, en particulier pour la localisation, la navigation et l'interaction avec l'environnement. Or les données perçues par les systèmes robotiques sont souvent complexes et sujettes à des imprécisions importantes. Pour remédier à ces problèmes, l'approche multi-capteurs utilise soit plusieurs capteurs de même type pour exploiter leur redondance, soit des capteurs de types différents pour exploiter leur complémentarité afin de réduire les imprécisions et les incertitudes sur les capteurs.

La validation de cette approche de fusion de données pose deux problèmes majeurs. Tout d'abord, le comportement des algorithmes de fusion est difficile à prédire, ce qui les rend difficilement vérifiables par des approches formelles. De plus, l'environnement ouvert des systèmes robotiques engendre un contexte d'exécution très large, ce qui rend les tests difficiles et coûteux.

L'objet de ces travaux de thèse est de proposer une alternative à la validation en mettant en place des mécanismes de tolérance aux fautes : puisqu'il est difficile d'éliminer toutes les fautes du système de perception, on va chercher à limiter leurs impacts sur son fonctionnement.

Nous avons étudié la tolérance aux fautes intrinsèquement permise par la fusion de données en analysant formellement les algorithmes de fusion de données, et nous avons proposé des mécanismes de détection et de rétablissement adaptés à la perception multi-capteurs. Nous avons ensuite implémenté les mécanismes proposés pour une application de localisation de véhicules en utilisant la fusion de données par filtrage de Kalman. Nous avons finalement évalué les mécanismes proposés en utilisant le rejeu de données réelles et la technique d'injection de fautes, et démontré leur efficacité face à des fautes matérielles et logicielles.

Mots Clés : *Sûreté de fonctionnement, Tolérance aux fautes, Injection de fautes, Perception, fusion de données.*

Abstract

Perception is a fundamental input for robotic systems, particularly for positioning, navigation and interaction with the environment. But the data perceived by these systems are often complex and subject to significant imprecision. To overcome these problems, the multi-sensor approach uses either multiple sensors of the same type to exploit their redundancy or sensors of different types for exploiting their complementarity to reduce the sensors inaccuracies and uncertainties.

The validation of the data fusion approach raises two major problems. First, the behavior of fusion algorithms is difficult to predict, which makes them difficult to verify by formal approaches. In addition, the open environment of robotic systems generates a very large execution context, which makes the tests difficult and costly.

The purpose of this work is to propose an alternative to validation by developing fault tolerance mechanisms : since it is difficult to eliminate all the errors of the perceptual system, We will try to limit impact in their operation.

We studied the inherently fault tolerance allowed by data fusion by formally analyzing the data fusion algorithms, and we have proposed detection and recovery mechanisms suitable for multi-sensor perception, we implemented the proposed mechanisms on vehicle localization application using Kalman filtering data fusion. We evaluated the proposed mechanims using the real data replay and fault injection technique.

Key Words : *Dependability, Fault tolerance, Fault injection, Perception, Data fusion.*

Introduction

La perception est une des entrées fondamentales de tout système robotique, en particulier pour la localisation, la navigation et l'interaction avec l'environnement. Toutefois, les données perçues par ces systèmes sont complexes et très souvent sujettes à des erreurs importantes. Pour remédier à ces problèmes l'approche multi-capteurs utilise soit plusieurs capteurs de même type pour exploiter leur redondance, soit des capteurs de types différents pour exploiter leur complémentarité afin de filtrer les bruits, éliminer certaines données aberrantes, augmenter la précision de la perception et obtenir une entrée correcte utilisable par le système. Mais multiplier les capteurs et les algorithmes de fusion de données sous-jacents augmente par conséquent les risques de fautes matérielles et logicielles. Par ailleurs, la validation de cette approche se heurte à deux problèmes majeurs :

- Tout d'abord, les algorithmes de fusion font partie du paradigme de programmation déclarative, qui consiste en une description d'un problème et d'un système afin d'atteindre une décision. Ce paradigme est souvent utilisé dans les applications d'intelligence artificielle telles que la planification, mais il est plus difficile à comprendre et à valider que la programmation impérative (qui est la description successive des étapes à exécuter). En général, les approches de programmation déclarative sont aujourd'hui interdites dans la norme de systèmes critiques. Par exemple, la norme ferroviaire 50l28 [cen, 2011] stipule que les logiciels d'intelligence artificielle ne sont pas recommandés dans les applications critiques, alors que la programmation procédurale (qui fait partie du paradigme impératif) est fortement recommandée. Ainsi, le comportement des algorithmes de fusion est difficile à prédire, ce qui les rend difficile à valider par des approches formelles, comme le modèle formel et la vérification par la preuve.

- Deuxièmement, l'environnement ouvert auquel est confronté le système robotique implique un contexte d'exécution quasi-illimité. En effet, dans des situations routières, les obstacles peuvent apparaitre à tout moment et de différentes manières, les conditions d'éclairage et de conduite peuvent varier,etc. Ainsi, la validation des systèmes automobiles exige des milliers

d'heures de conduite sur les routes, avec aucune certitude que toutes les situations possibles ont été rencontrées. Pour cette raison, les tests sont une opération difficile, longue et coûteuse.

L'objectif de ces travaux de thèse est de proposer une solution à ces problèmes de validation, en utilisant un moyen de sûreté de fonctionnement complémentaire à l'élimination des fautes : la tolérance aux fautes. Ainsi, plutôt que de garantir par des tests et une validation exhaustive que le système de perception ne comporte aucune faute, on implémente des mécanismes pour assurer que l'activation de fautes dans le système n'entrave pas sa sûreté de fonctionnement. Puisqu'il est difficile d'éliminer toutes les fautes du système, on va chercher à limiter leurs impacts sur son fonctionnement. Le présent manuscrit est décomposé en six chapitres :

Dans le premier chapitre, nous allons d'abord présenter les concepts et les terminologies des deux domaines impliqués dans ces travaux de recherche, à savoir la sûreté de fonctionnement, en particulier les méthodes de tolérance aux fautes, et la fusion de données multi-capteurs. Nous nous intéressons ensuite aux différentes techniques de tolérance aux fautes en fusion de données proposées dans la littérature.

Dans le second chapitre nous présentons deux contributions pour assurer des services de tolérance aux fautes en fusion de données : la première applique la technique de duplication / comparaison et la seconde applique une étude formelle des algorithmes de fusion de données (sur un exemple de cas d'études utilisant les fonctions de croyance). Une comparaison qualitative de ces deux approches est ensuite réalisée.

Dans le troisième chapitre nous exposons un exemple d'application implémentant l'architecture de duplication / comparaison pour la localisation des robots mobiles, et nous détaillons les services de détection et de recouvrement de fautes assurés.

Le quatrième chapitre est consacré à une étude expérimentale de l'implémentation exposée en chapitre 3 et l'évaluation de ses performances en termes de sûreté de fonctionnement. Cette évaluation utilise des données réelles et la technique d'injection de fautes.

Le cinquième chapitre est consacré au rétablissement du système vis-à-vis de fautes logicielles. Il présente une application d'estimation de l'angle de lacet d'un robot mobile utilisant le filtre de Kalman. Il emploie la diversification fonctionnelle pour assurer le recouvrement d'une faute de développement pouvant exister dans les algorithmes de filtre de Kalman. Une validation des mécanismes proposés est détaillée.

Enfin, nous concluons en retraçant les points importants de ce mémoire, soulignant l'importance de cette problématique, rappelant les apports des mécanismes

proposés et identifiant leurs limitations. Finalement nous présentons plusieurs pistes de recherche qui pourraient améliorer et enrichir nos travaux dans le futur.

Terminologie et état de l'art

Sommaire

Les travaux de recherche de cette thèse portent sur deux domaines différents, chacun utilisant une terminologie et des concepts spécifiques. La sûreté de fonctionnement offre divers moyens, dont la tolérance aux fautes, pour faire face aux défaillances des systèmes informatiques. La fusion de données multi-capteurs fournit des méthodes et techniques pour combiner les sorties de capteurs afin de mieux percevoir l'environnement des systèmes robotiques. Les mécanismes de tolérance aux fautes et de fusion de données utilisent toutes deux la redondance ; cependant les premiers tentent de détecter et tolérer les fautes du système, tandis que les seconds se concentrent sur les aléas d'un environnement ouvert et les imprécisions de capteurs.

Ce chapitre présente les concepts de base de ces deux domaines. Il décrit d'abord les concepts liés à la sûreté de fonctionnement, en particulier la tolérance aux fautes, puis ceux liés à la fusion de données. Il propose enfin un état de l'art concernant les mécanismes de tolérance aux fautes en fusion de données, et il conclut par une analyse critique de ces derniers.

1.1 La sûreté de fonctionnement

Les systèmes informatiques sont aujourd'hui incontournables dans notre société, et de plus en plus présents dans notre environnement (aéronautique, automobile, énergie, téléphonie mobile, transport ferroviaire...). Ces systèmes, conçus pour assurer des fonctionnalités en temps réel, peuvent porter sur des applications

critiques comme des centrales nucléaires, des avions ou des trains, dont une défaillance peut avoir des conséquences catastrophiques en termes économiques ou de vies humaines. En réponse à cette préoccupation, la notion de sûreté de fonctionnement a été développée en fournissant des moyens pour assurer une confiance justifiée dans les systèmes informatiques.

1.1.1 Principes généraux de la sûreté de fonctionnement

La sûreté de fonctionnement est une notion générique qui mesure la qualité de service délivré par un système, de manière à ce que l'utilisateur ait en lui une confiance justifiée [Laprie et al., 1992][Avizienis et al., 2004]. La décomposition systématique des concepts de la sûreté de fonctionnement se fait selon trois critères (Figure 1.1) : *ses attributs*, les propriétés attendues du système ; *ses entraves*, les comportements inacceptables du système qui sont causes ou conséquences de la non sûreté de fonctionnement ; *ses moyens*, les méthodes qui permettent à un système d'être apte à accomplir correctement sa fonction en plaçant une confiance justifiée dans le service qu'il délivre.

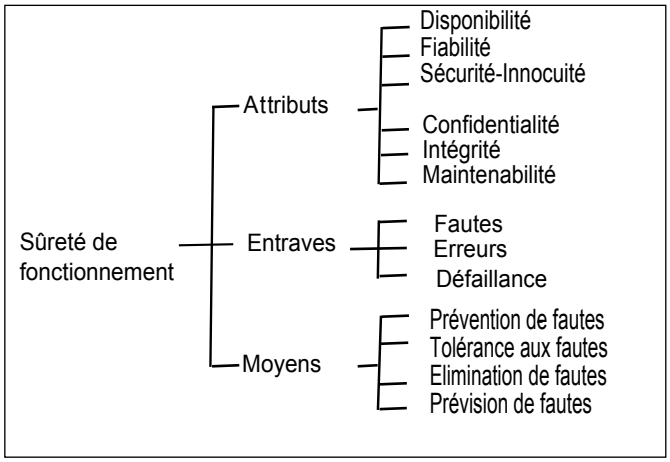

Figure 1.1 – Arbre de la sûreté de fonctionnement

Attributs

Les attributs de la sûreté de fonctionnement correspondent aux propriétés que doit vérifier un système. Une importance plus au moins grande est accordée à ces différents attributs selon l'application et l'environnement auxquels est destiné le système informatique considéré. Six attributs de la sûreté de fonctionnement sont définis :

- *La disponibilité* : le fait qu'un système soit prêt à l'utilisation et que rien n'entrave sa fonction.

- *La fiabilité ou continuité du service* : la délivrance continuelle et correcte du service requis.
- *La sécurité-innocuité* : l'absence de conséquences catastrophiques pour l'environnement et l'utilisateur du système.
- *La maintenabilité* : l'aptitude d'un système à être remis rapidement dans un état opérationnel dans lequel il peut accomplir une fonction requise.
- *La sécurité-confidentialité* : l'absence de divulgation non autorisée de l'information.
- *L'intégrité* : l'absence d'altérations inappropriées de l'information.

Dans ce manuscrit, nous nous intéressons en particulier à la fiabilité, qui caractérise la continuité de service, et la sécurité-innocuité, qui caractérise la non occurrence de conséquences catastrophiques pour le système et son environnement. Notons qu'essayer d'atteindre ces deux attributs peut être contradictoire. En effet, pour un véhicule autonome dans une situation dangereuse, la sécurité peut exiger d'arrêter et d'évaluer la situation, tandis que la fiabilité demanderait de poursuivre le trajet.

Entraves

Les entraves à la sûreté de fonctionnement représentent les comportements indésirables du système. Elles sont de trois types : *les fautes*, *les erreurs* et *les défaillances*, comme le montre l'exemple de la figure 1.2. Elles sont liées par la relation de causalité suivante : une faute est la cause adjugée ou supposée d'une erreur, et l'erreur est susceptible d'entraîner une défaillance (c'est-à-dire que le service délivré par le système diverge du service attendu) comme le montre la Figure 1.2.

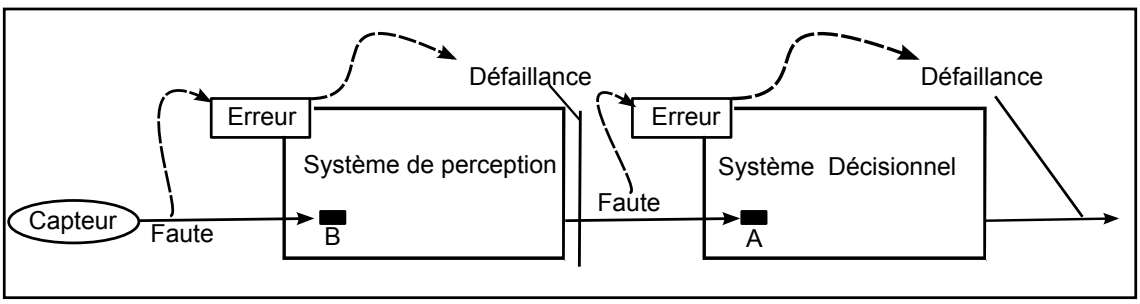

Figure 1.2 – Exemple de propagation d'erreur dans un système de perception

Sur la Figure 1.2, le système décisionnel A prend en entrée la sortie du système de perception B. Le service correct de A dépend ainsi du service correct de B. Pour

A, la défaillance de B est considérée comme une faute externe, qui à son tour peut provoquer une erreur interne dans A, et potentiellement sa défaillance.

Moyens

Pour assurer la sûreté de fonctionnement d'un système, l'utilisation conjointe d'un ensemble de méthodes est nécessaire ; ces techniques sont regroupées dans les quatre moyens suivants :

- *Prévention de fautes* : méthodes qui empêchent l'occurrence ou l'introduction de fautes dans le système ;
- *Tolérance aux fautes* : méthodes qui permettent à un système de remplir correctement sa fonction en dépit de fautes ;
- *Élimination des fautes* : méthodes qui réduisent la présence (le nombre et la gravité) des fautes dans un système ;
- *Prévision des fautes* : méthodes qui estiment la présence, la création et les conséquences des fautes dans un système.

Ces méthodes peuvent être classées en deux approches complémentaires [Lussier, 2007] pour concevoir un système sûr de fonctionnement :

- l'approche d'évitement de fautes englobe la prévention et l'élimination de fautes cherchant à concevoir un système sujet au moins de fautes possibles,
- l'approche d'acceptation des fautes englobe la tolérance aux fautes et la prévision des fautes : comme il y a toujours des fautes qu'on ne peut pas éviter, on essaie d'évaluer leurs impacts sur le système et de réduire la sévérité des défaillances qu'elles peuvent causer (si possible jusqu'à la suppression des défaillances).

1.1.2 La tolérance aux fautes

Dans ce manuscrit, nous nous concentrons sur les approches de tolérance aux fautes. Cette section en décrit les notions et les méthodes.

1.1.2.1 Principe de tolérance aux fautes

La mise en œuvre de la tolérance aux fautes s'effectue généralement en deux étapes : la *détection d'erreurs* et le *rétablissement du système* :

- **La détection d'erreurs** constitue un pré-requis quasiment indispensable à la mise en œuvre de solutions de tolérance aux fautes. Les formes les plus courantes de détection d'erreurs sont *la duplication et comparaison*, *le contrôle temporel*, et *le contrôle de vraisemblance*.

  - ◇ *La duplication et comparaison* compare les résultats fournis par au moins deux unités redondantes indépendantes vis-à-vis des fautes à tolérer et fournissant le même service. Typiquement les fautes physiques sont détectables par une redondance des composants matériels tels que les capteurs dans les systèmes de perception, et les fautes logicielles par une programmation N-version telle que la diversification des algorithmes de fusion de données.

  - ◇ *Le contrôle temporel* détecte des erreurs temporelles dans le système, par exemple en contrôlant que son temps de réponse ne dépasse pas une valeur maximale (time out). Dans un système de perception multi-capteurs, on peut de cette façon surveiller l'activité des capteurs en utilisant un chien de garde.

  - ◇ *Le contrôle de vraisemblance* cherche à détecter des erreurs en valeurs aberrantes pour le système. Un exemple typique de contrôle de vraisemblance pour un capteur de perception est de vérifier si sa sortie appartient à une fourchette de valeurs possibles.

- **Le rétablissement du système** transforme l'état erroné détecté du système en un état jugé exempt d'erreurs. Il est effectué par deux méthode complémentaires :

  - ◇ *Le traitement d'erreur* permet de substituer un état exempt d'erreurs à l'état erroné. Cette substitution peut elle-même prendre trois formes : *la reprise* qui ramène le système dans un état correct survenu avant l'occurrence de l'erreur ; *la poursuite* qui cherche un nouvel état à partir duquel le système peut fonctionner (éventuellement en mode dégradé) ; et *la compensation* où l'état erroné comporte suffisamment de redondance pour permettre sa transformation en un état correct.

    - ✓ *La reprise* consiste à revenir vers un état antérieur correct. Cela nécessite de capturer et sauvegarder périodiquement l'état du système, en créant un ensemble de points de reprise (ou checkpoints) possibles, à partir desquels un état cohérent peut être restauré. L'avantage de cette technique est d'être générale et indépendante

de l'application. Cependant, plusieurs difficultés lui sont liées : les points de reprise doivent eux-mêmes être exempts de fautes, et ils sont coûteux en termes de taille des sauvegardes. Il faut encore ajouter à cela la difficulté d'effectuer des sauvegardes cohérentes, et le sur-coût temporel nécessaire à leur établissement.

- ✓ *La poursuite* nécessite de trouver un nouvel état correct à partir duquel le système peut fonctionner, généralement en mode dégradé. Cette technique reste spécifique et dépendante de l'application. La recherche d'un nouvel état acceptable consiste souvent en une ré-initialisation du système et en l'acquisition d'un nouveau contexte d'exécution auprès de l'environnement.
- ✓ *La compensation* nécessite la présence de redondances dans le système lui permettant de fournir un service correct malgré les erreurs qui pourront l'affecter. Cette compensation se présente sous deux formes : *détection et compensation* si elle est consécutive à une détection d'erreur, ou *masquage d'erreur* si elle est appliquée systématiquement sans détection d'erreurs.

◇ *Le traitement de fautes* empêche une nouvelle activation de la faute à l'origine de l'erreur. Il est possible soit d'apporter une solution réparatrice, soit de se priver des composants atteints par la faute pour qu'elle ne s'active plus.

Cette méthode est composée de quatre phases successives :

- ✓ *Le diagnostic* identifie et localise la faute activée responsable de l'état erroné du système ;
- ✓ *L'isolement* rend la faute dormante en excluant la participation du composant erroné de la délivrance du service, par moyen physique ou logiciel ;
- ✓ *La reconfiguration* compense l'isolement du composant défaillant soit en basculant sur des composants redondants, soit en réassignant ses tâches à d'autres composants ;
- ✓ *La ré-initialisation* vérifie et met à jour la nouvelle configuration du système.

1.1.2.2 Validation des mécanismes de tolérance aux fautes

Étant donné que les mécanismes de tolérance aux fautes emploient des redondances matérielles et logicielles, elles engendrent un sur-coût en termes financier et en temps

de développement. Il est donc indispensable de procéder à l'évaluation de leurs performances pour connaître leur apport en termes de sûreté de fonctionnement par rapport à ces divers coûts. De plus il est nécessaire de s'assurer que les mécanismes n'introduisent pas de nouvelles fautes dans le système. Pour répondre à ces problèmes, l'évaluation et la validation des mécanismes de tolérance aux fautes peuvent être réalisées suivant deux approches complémentaires : la vérification formelle et l'injection de fautes.

- *La vérification formelle* consiste à utiliser des techniques formelles comme l'analyse statique, la preuve mathématique, ou l'analyse de comportement, pour obtenir des certitudes sur le comportement du système. Ces méthodes permettent également de valider le comportement d'un système en présence de fautes, à condition de disposer de modèles formels décrivant les fautes considérées et leurs conséquences sur le comportement du système.

- *L'injection de fautes* est une technique de validation et d'évaluation des mécanismes de tolérance aux fautes. Elle consiste à réaliser des expériences contrôlées par l'introduction volontaire de fautes dans un système ou dans un modèle du système, et à observer son comportement [Arlat et al., 1993]. L'objectif est de comparer le comportement nominal du système (sans injection de faute), avec son comportement en présence de fautes injectées pendant l'exécution afin de révéler les déficiences des mécanismes de tolérance aux fautes implémentés.

  On peut simuler des fautes pour représenter des fautes physiques ou logicielles, par altération du contenu des composants constituant le système, ou par modification des entrées et des sorties d'un composant. Un exemple de fautes physiques dans un système de perception consiste par exemple à altérer et modifier les sorties des capteurs.

1.2 Fusion de données multi-capteurs

Après avoir introduit dans la section précédente les concepts de base de la sûreté de fonctionnement, nous présentons ici le domaine de la fusion de données, qui joue un rôle fondamental pour les systèmes de perception multi-capteurs. Nous présentons d'abord ses concepts de base, puis nous exposons ses différents formalismes, en particulier la fusion de données par théorie des fonctions de croyance, et par filtre de Kalman.

1.2.1 Principes généraux de la fusion de données

Les systèmes de fusion de données sont aujourd'hui largement utilisés dans divers domaines. Les premières applications employant les concepts de la fusion de données ont été développées dans le domaine militaire [White, 1991] [Hall and McMullen, 2004] [Hall and Llinas, 1997] [Rouchouze, 1994], et elles restent encore d'actualité. Au cours des dernières années, la fusion de données a été adaptée et développée pour des applications dans d'autres domaines tels que la perception satellite, l'imagerie aérienne ou radar, la robotique [Yi et al., 2000], et l'intelligence artificielle.

Plusieurs sens sont donnés à la fusion de données, ou fusion d'informations. Nous reprenons ici la définition proposée par [Bloch and Maître, 1997] : La fusion d'informations consiste à combiner des informations issues de plusieurs sources afin d'améliorer la prise de décision. Nous entendons par source d'informations, tout système, des capteurs physiques à l'informateur humain, observant la situation réelle ou fournissant une information a priori sur les événements possibles. D'autres définitions bien détaillées de ces concepts peuvent être trouvées dans [Boström et al., 2007].

En fusion de données multi-capteurs, les entrées de plusieurs capteurs sont combinées pour former une représentation complète de l'environnement observé : *le modèle de l'environnement*. La fusion de données cherche à tirer profit de toutes les informations disponibles sur un problème donné pour contrer les imperfections [Bloch, 2003] de chacune des sources. On donne ici les définitions de ces différentes imperfections [Bradai, 2007]

- *l'imprécision* : elle concerne le contenu de l'information, et mesure donc un défaut quantitatif de connaissance sur une mesure.

- *l'incertitude* : c'est une notion relative à la vérité d'une information, et mesurant un défaut qualitatif de connaissance sur une mesure. Elle caractérise le degré de conformité de l'information par rapport à la réalité ou encore l'assurance d'une source en l'information fournie.

- *l'incomplétude* : elle est le manque d'information apporté par la source. Cette notion est importante en perception car, par nature, les capteurs de perception ont une vue partielle de l'environnement.

- *l'ambiguïté* : c'est le risque qu'une information à conduire à deux interprétations. Un des objectifs de la fusion est de lever les ambiguïtés d'une source grâce aux informations apportées par les autres sources ou par les connaissances supplémentaires.

- *le conflit* : il caractérise le fait que deux ou plusieurs informations conduisent à des interprétations contradictoires, et donc incompatibles. Les situations conflictuelles sont fréquentes dans les processus de fusion de données, et posent toujours des problèmes difficiles à résoudre.

La fusion de données peut également être utilisée pour résoudre des problèmes concernant l'hétérogénéité des sources d'information. Les capteurs utilisés peuvent cibler les mêmes informations de l'environnement (capteurs compétitifs), ou des informations complémentaires (capteurs coopératifs). Ces stratégies sont détaillées en section 1.2.1.2.

1.2.1.1 Étape de la Fusion de données

La mise en œuvre d'un processus de fusion de données comprend quatre étapes principales :

- *Modélisation* : Le choix d'un formalisme mathématique pour représenter l'information provenant de chaque source.
- *Estimation* : Étape dépendante de la modélisation. Elle n'est pas systématique mais souvent nécessaire pour la plupart des formalismes. Des informations supplémentaires peuvent aussi intervenir à ce niveau. Il s'agit par exemple de l'estimation des masses de croyances dans la fusion de données par fonctions de croyance.
- *Combinaison* : Le choix d'un opérateur compatible avec le formalisme adopté dans le but de combiner les différentes informations.
- *Décision* : Le choix d'un critère de décision sur le résultat de la combinaison des informations.

1.2.1.2 Stratégies de configuration

En fusion de donnée multicapteurs, les capteurs peuvent être intégrés dans le système de perception selon trois stratégies [Durrant-Whyte, 1988] [Frémont, 2009] [Khaleghi et al., 2009] :

- *Coopérative* : la combinaison de plusieurs capteurs permet d'obtenir une information indisponible avec chaque capteur pris individuellement. Par exemple une combinaison de deux lidars à l'avant et l'arrière du véhicule fournit une détection d'obstacle tout autour du véhicule.

- *Compétitive* : chaque capteur réalise la même fonction, le but étant alors de réduire les effets des données incertaines et erronées. Par exemple la fusion d'un odomètre et d'un GPS pour la localisation d'un véhicule réduit les incertitudes et les erreurs de positionnement.

- *Complémentaire ou redondante* : les capteurs ne dépendent pas directement les uns des autres et pourrait offrir un service acceptable seul, mais une fois combinés ils améliorent l'image et la précision du phénomène observé. Comme par exemple l'utilisation d'un GPS pour la localisation combiné avec un télémètre laser pour la construction d'une carte d'environnement pour les véhicules intelligents.

1.2.1.3 Formalismes de fusion de données

La mise en œuvre d'un mécanisme de fusion de données est généralement appliquée dans l'un des trois grands cadres théoriques, à savoir la théorie des probabilités [Grandin, 2006] [1], la théorie des possibilités [Dubois and Prade, 1988a], [Dubois and Prade, 1994] et la théorie des fonctions de croyance [Dempster, 1967], [Shafer, 1976]. Cette section a pour but de donner un aperçu globale des deux premières théories.

Théorie des probabilités

La théorie des probabilités est l'une des théories utilisées dans le domaine de la fusion de données [Mitchell, 2007]. Cette approche repose sur un cadre mathématique solide éprouvé par de nombreuses études depuis des années. Dans le cadre probabiliste, les imperfections de l'information et les informations elles mêmes sont modélisées à partir de distributions de probabilités. De ce fait, cette approche ne permet de modéliser que l'incertitude de l'information.

Comme cette théorie raisonne sur des singletons qui représentent les différentes solutions, l'espace de définition doit être exhaustif et exclusif. Ceci entraîne que le monde doit être fermé, hypothèse qui ne correspond pas toujours à la réalité [Martin, 2005]. L'exclusivité provient de l'hypothèse d'additivité des probabilités. En outre, elle ne permet pas de modéliser la méconnaissance (le fait que l'on ne sait pas), ni de gérer les conflits entre les sources. Parmi les techniques de fusion de données utilisant cette théorie en robotique, on trouve les méthodes d'estimation basées sur le filtre de Kalman qui est largement utilisé pour différentes applications comme la navigation, la localisation, le suivi de cible. Ces méthodes de filtrage de Kalman sont détaillées dans la section 1.2.2.

[1] URL http ://www.techniques-ingenieur.fr

Théorie des possibilités

La théorie des possibilités [Dubois and Prade, 2000] est un outil mathématique beaucoup plus récent que la théorie des probabilités. Elle permet une gestion efficace de l'imprécision et de l'incertitude qui peut être liée à certaines données. Elle a été proposé par L Zadeh en 1987 en s'appuyant sur la théorie des sous ensembles flous qu'il a inventée en 1965. Elle a été développée par la suite en France par Dubois et Prade pour traiter dans un seul cadre l'incertitude et l'imprécision inhérente à certaines données.

L'outil de base de la théorie des possibilités est la distribution de possibilités qui est équivalente à un degré d'appartenance (ou fonction d'appartenance) de la théorie des sous-ensembles flous.

La théorie des possibilités modélise la préférence que l'on a pour une proposition, c'est-à-dire un moyen de dire dans quelle mesure la réalisation d'un événement est possible et dans quelle mesure on en est certain. Ces deux évaluations sont formalisées à travers une mesure de possibilité et une mesure de nécessité respectivement.

Pour avoir une vue d'ensemble des deux formalismes probabiliste et possibiliste, et leurs étapes de fusion le lecteur peut se référer à [Bradai, 2007] et [Martin, 2005].

Théorie des fonctions de croyances

La théorie des fonctions de croyance, également connue sous le nom de théorie de l'évidence ou théorie de Dempster-Shafer, est issue des travaux de Dempster en 1967 [Dempster, 1967], repris par Shafer [Shafer, 1976] sous le nom de théorie de l'évidence. Cette théorie peut être vue comme un sur-ensemble de plusieurs approches quantitatives [Démotier et al., 2006] à savoir : la théorie des probabilités, la théorie des possibilités présentées ci dessus, la logique floue, et d'autres. La théorie des fonctions de croyance peut être considérée comme plus générale [Masson, 2005] que celle des probabilités ou des possibilités puisque l'on retrouve celles-ci comme des cas particuliers, et elles sont (partiellement) des branches spéciales de la théorie de l'évidence qui sert d'interconnexion entre elles [Lohweg et al., 2011]. La Figure 1.3 illustre symboliquement l'interconnexion de ces différentes approches et leurs lien avec la théorie des fonctions de croyance.

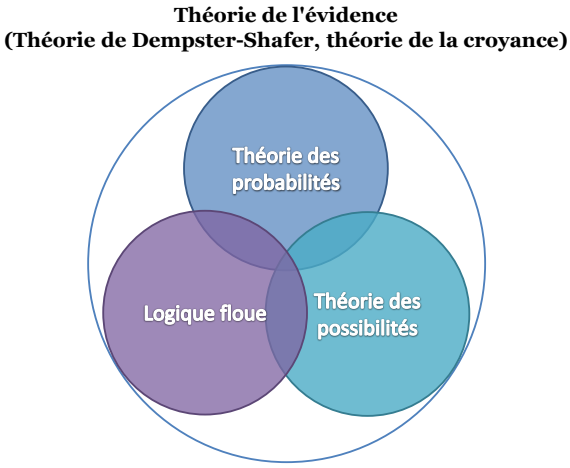

Figure 1.3 – Théories intégrées dans la théorie des fonctions de croyance.

1.2.2 Fusion de données par filtre de Kalman

Le filtre de Kalman a été inventé et publié par R.E. Kalman en 1960 [Kalman et al., 1960] pour le cas discret et repris par Kalman et Bucy [Kalman and Bucy, 1961] pour le cas continu. Cette méthode a été largement appliquée dans de nombreuses applications robotiques [Freeston, 2002] [Hu et al., 2003] [Casanova et al., 2008] [Kim et al., 2007] (comme la navigation autonome, suivi de cible et localisation). Les algorithmes de fusion à base de filtre de Kalman consistent à estimer de manière optimale et de façon récurrente l'état d'un système inconnu. Ce résultat est obtenu grâce à une série de calculs récurrents pour une meilleure estimation des variables d'état du système avec une correction proportionnelle à l'erreur entre une étape de prédiction et celle de correction. C'est une technique de fusion de données probabiliste.

1.2.2.1 Filtre de Kalman linéaire

Le filtre de Kalman suppose que l'état du système et les mesures des capteurs peuvent être décrits par un système dynamique linéaire. Il s'agit de deux modèles d'équations linéaires : l'un modélise l'évolution de l'état du système au fil du temps et l'autre décrit comment les mesures sont liées à cet état. Le filtre de Kalman suppose un modèle linéaire, adéquat pour le problème à modéliser. Lorsque le problème n'est pas linéaire, alors nous pouvons utiliser des techniques de linéarisation pour transformer un problème non-linéaire en un problème linéaire, en utilisant le filtre de Kalman étendu (EKF).

- **Modélisation :** Le système d'équations utilisé dans le filtre de Kalman repose sur la définition de deux modèles qui sont le *modèle du processus* et le *modèle*

d'observations :

◇ *le modèle du processus* décrit comment le vrai état du système évolue au fil du temps. Le filtre de Kalman a besoin de ce modèle afin de faire des prédictions sur cet état. Il suppose que l'état du système évolue en fonction de l'équation linéaire 1.1

$$x_k = A_k x_{k-1} + B_k u_{k-1} + w_k \tag{1.1}$$

- ✓ $x_{k-1}, x_k \in R^n$ sont les vecteurs d'état du système a l'instant $k-1$ et k.
- ✓ $A_k \in R^{nXn}$ est la matrice de transition qui fait le lien entre les vecteurs d'état du système à deux étapes successives.
- ✓ $u_{k-1} \in R^m$ est le vecteur de commande, il caractérise l'entrée du système.
- ✓ $B_k \in R^{mXm}$ est la matrice de commande, elle fait le lien entre la commande et l'état du système.
- ✓ $w_k \in R^n$ représente le bruit de modèle du système, supposé blanc, centré et de matrice de covariances Q_k connue et constante.

$$E(w_k) = 0 \text{ et } var(w_k) = Q_k$$

◇ *le modèle d'observation* décrit comment les mesures sont liées à l'état. Le filtre de Kalman a besoin de ce modèle afin de corriger la prédiction d'état quand une mesure est disponible. Le filtre de Kalman suppose que les mesures peuvent être modélisées par une équation linéaire 1.2 liant l'état du système à une mesure,

$$z_k = H_k x_k + v_k \tag{1.2}$$

- ✓ $z_k \in R^p$: est le vecteur de mesures à l'instant k qui dépend linéairement de l'état du système x_k
- ✓ $H_k \in R^{pXn}$: est la matrice d'observation liant l'état du système x_k aux mesures z_k
- ✓ v_k est le bruit de mesures, supposé blanc, centré et de matrices de covariances R_k supposée connue et constante.

$$E(v_k) = 0 \;\; var(v_k) = R_k$$

- **Caractéristiques des bruits :** Comme mentionné et modélisé ci dessus, le système et les capteurs sont soumis à des bruits. Le filtre de Kalman suppose que le bruit du système w_k dans (1.1), et le bruit de mesure v_k dans (1.2) sont des variables aléatoires indépendantes, gaussiennes et de moyennes nulles. Avec ces hypothèses, nous pouvons écrire la façon dont ces bruits sont distribués. Si nous posons $Q_k = E[(w_k)(w_k)^T]$ la covariance de bruit de modèle du système , et $R_k = E[(v_k)(v_k)^T]$ la covariance de bruit de mesures à l'instant k, nous pouvons exprimer le bruit de système w_k et celui de mesures v_k comme suit :

$$w_k \sim N(0, Q_k) \tag{1.3}$$

$$v_k \sim N(0, R_k) \tag{1.4}$$

où $N(\mu, \Sigma)$ désigne la fonction gaussienne de moyenne μ et covariance Σ. La diagonale principale des matrices de covariance Q_k et R_k contient la variance des vecteurs d'état et de mesures, respectivement. Les autres éléments de ces deux matrices sont nuls, puisque nous supposons que les bruits sont indépendants.

Notons que cette caractéristique du bruit est généralement déterminée en pratique de manière empirique, à travers une détermination expérimentale des matrices Q, et R.

- **Processus de filtre de Kalman :** Le filtre de Kalman estime un état d'un système en utilisant une forme récurrente. Cette dernière englobe deux étapes : l'estimation a priori (*la prédiction*) et l'estimation a posteriori (*la correction*). Les paramètres du système sont estimés premièrement à partir des valeurs de l'étape précédente et ils sont ensuite corrigés par des mesures dans l'étape de correction.

  Les étapes constituant le filtrage de Kalman dans un cadre linéaire sont :

  - ◇ *Prédiction* : Cette étape permet la prédiction de l'état et de sa précision à l'instant k à partir de l'état à l'instant $k-1$ et du modèle d'évolution du système. On obtient donc l'état $\hat{x_k}^-$ avec sa matrice de covariances associée P_k^- :

$$\hat{x}_k^- = A_k \hat{x}_{k-1}^+ + B_k u_{k-1} \tag{1.5}$$

$$P_k^- = A_k P_{k-1}^+ A_k^T + Q_k \tag{1.6}$$

  - ◇ *Correction :* Une fois la mesure z_k disponible, l'état prédit peut alors être

corrigé par l'innovation $\hat{S}_k$(et sa covariance associée 1.7) pondérée par le gain du filtre k_k. On en déduit alors l'estimation de l'état $\hat{x_k}^+$ avec sa matrice de covariances associée P_k^+.

$$\hat{S}_k = z_k - H_k.\hat{x}_k^- \tag{1.7}$$

$$S_k = H_k.P_k^-.H_k^T + R_k \tag{1.8}$$

$$k_k = P_k^-.H_k^T.S_k^{-1} \tag{1.9}$$

$$\hat{x}_k^+ = \hat{x}_k^- + k_k.\hat{S}_k \tag{1.10}$$

$$P_k^+ = P_k^- - k_k.H_k.P_k^- \tag{1.11}$$

Le résidu (innovation) $\hat{S}_k$ reflète l'écart entre la mesure prédite et la mesure réelle. Un résidu de valeur zéro, signifie que les deux mesures sont égales.

1.2.2.2 Filtre de Kalman étendu

Le filtre de Kalman étendu fournit une méthode appliquant la technique de filtre de Kalman pour un problème non linéaire par la linéarisation de l'estimation autour de l'estimée actuelle à l'aide des dérivées partielles.

- **Modélisation :** pour le filtre de Kalman étendu, le système et l'observation sont modélisés par des equations non linéaires.

  ◇ *Modèle du processus* pour le filtre de Kalman étendu, le processus est régi par des équations différentielles stochastiques non linéaires.

  $$x_k = f(x_{k-1}, u_{k-1}) + w_{k-1} \tag{1.12}$$

  ◇ *Modèle d'observations* L'équation de mesure pour le filtre de Kalman étendu est considérée comme une fonction non-linéaire des états.

  $$z_k = h(x_{k-1}) + v_{k-1} \tag{1.13}$$

  f et h dans les equations 1.12 et 1.13 respectivement sont des fonctions non-linéaires connues, u_k est le vecteur de commande et w_k et v_k sont des processus de type bruit blanc non corrélés. La non-linéarité peut être soit dans le modèle du processus soit dans le modèle de mesure, soit dans les deux à la fois.

- **Processus de filtre de Kalman étendu :** Pour appliquer le filtre de Kalman, les équations non linéaires d'état du système et de mesures sont linéarisées avec une approximation du premier ordre en utilisant la matrice Jacobienne.

  ◇ F est la matrice Jacobienne des dérivées partielles de f par rapport à x, qui est :

  $$F = \left.\frac{\partial f}{\partial x}\right|_{x=\hat{x}} \tag{1.14}$$

  ◇ H est la matrice Jacobienne des dérivées partielles de h par rapport à x, qui est :

  $$H = \left.\frac{\partial h}{\partial x}\right|_{x=\hat{x}} \tag{1.15}$$

  La propagation de l'état du système de l'instant $k-1$ a k peut être représentée par les équations suivantes :

  $$\hat{x}_k^- = f(\hat{x}_{k-1}^+) + w_k \tag{1.16}$$

  $$P_k^- = F P_{k-1}^+ F^T + Q_k \tag{1.17}$$

  Comme précédemment, lorsque la mesure z_k est disponible à l'instant k, l'estimation est mise à jour par les équations suivantes :

  $$\hat{S}_k = z_k - H_k.\hat{x}_k^- \tag{1.18}$$

  $$S_k = H_k.P_k^-.H_k^T + R_k \tag{1.19}$$

  $$k_k = P_k^-.H_k^T.S_k^{-1} \tag{1.20}$$

  $$\hat{x}_k^+ = \hat{x}_k^- + k_k.\hat{S}_k \tag{1.21}$$

  $$P_k^+ = P_k^- - k_k.H_k.P_k^- \tag{1.22}$$

1.2.3 Fusion de données par fonction de croyance

La théorie des fonctions de croyance permet à la fois la modélisation et l'utilisation de données incertaines et imprécises, ainsi que de données qualitatives et quantitatives. Cette théorie est bien connue pour prendre en compte ce qui reste inconnu, tout autant que ce qui est déjà connu.

Cette section présente les différentes étapes de fusion de données par fonctions de croyance.

Modélisation : La théorie des fonctions de croyances repose sur la manipulation de fonctions définies sur des sous ensembles et non sur des singletons comme dans la théorie des probabilités. Ces fonctions sont habituellement définies dans l'intervalle [0,1], et sont appelées fonctions de masse, ou encore masses de croyance. La théorie des fonctions de croyances nécessite la définition d'un ensemble Ω appelé cadre de discernement. Cet ensemble est constitué de toutes les hypothèses (toutes les décisions possibles sur la fusion de données).

- **Fonction de masse :** Soit ω une variable définie sur un domaine fini Ω appelé cadre de discernement supposé exhaustif (hypothèse du monde fermé : la solution est dans Ω) et exclusif (la solution est unique). La croyance d'une source concernant la valeur prise par ω est représentée par une masse qui est une fonction de l'ensemble 2^Ω des parties de Ω dans [0,1], telle qu'on ait : $\sum_{\lambda \subseteq \Omega} m(\lambda) = 1$, avec la quantité $m(\lambda)$ représentant la croyance que l'on met exactement sur la proposition λ.

  Les éléments F tels que $m(F) \neq 0$ sont appelés les *éléments focaux*. A la différence de la théorie des probabilités on repartit la masse non pas sur Ω mais sur 2^Ω. C'est ce principe qui permet à la théorie des fonctions de croyances de modéliser les imprécisions. En effet, soient C et D deux éléments de Ω, $m(\{C, D\})$ représente la croyance de la source que la proposition C ou la proposition D est vraie, sans savoir laquelle est correcte.

  Exemple 1.1. *Soit ω l'état d'une porte qui pourra prendre deux valeurs* (Ouvert ou Fermé).
  $\Omega = \{\text{Ouvert}, \text{Fermé}\}$, *et* $2^\Omega = \{\{\text{Ouvert}\}, \{\text{Fermé}\}, \{\text{Ouvert}, \text{Fermé}\}, \emptyset\}$.
  On peut par exemple définir la masse m telle que $m(\{\text{Ouvert}\}) = 0.2$, $m(\{\text{Fermé}\}) = 0.5$, et $m(\Omega) = 0.3$.

- **La fonction de crédibilité** représente la croyance minimale d'une source définie à partir des masses élémentaires de croyance portées par les éléments focaux. Pour $A \subseteq \Omega$, $Bel(A)$ correspond à la somme des masses données aux sous ensembles non vides de A. C'est donc la croyance totale allouée à A par la connaissance disponible.

$$Bel(A) = \sum_{\emptyset \neq B \subseteq A} m(B) \quad \forall A \subseteq \Omega \tag{1.23}$$

  Exemple 1.2. *D'après l'exemple 1.1, $Bel(\{\text{Ouvert}\}) = m(\{\text{Ouvert}\}) = 0.2$.*

- **La plausibilité** (toujours plus grande que la crédibilité) est la fonction duale de la fonction de crédibilité ; elle mesure le degré maximal susceptible d'être

alloué à A. Pour $A \subseteq \Omega$, $Pl(A)$ correspond à la somme des masses données aux sous ensembles non vides qui intersectent A.

$$Pl(A) = \sum_{A \cap B \neq \emptyset} m(B) \quad \forall A \subseteq \Omega \tag{1.24}$$

Exemple 1.3. *D'après l'exemple 1.1,* $Pl(\{\text{Ouvert}\}) = m(\{\text{Ouvert}\}) + m(\{\text{Ouvert}, \text{Fermé}\}) = 0.2 + 0.3 = 0.5$.

L'incertitude sur un sous-ensemble $A \subseteq \Omega$ est bornée par l'intervalle $[Bel(A); Pl(A)]$. La Figure 1.4 [Aquirre-Martinez, 2013] montre une explication graphique des fonctions de crédibilité, et de plausibilité et leur relation avec la fonction de masse.

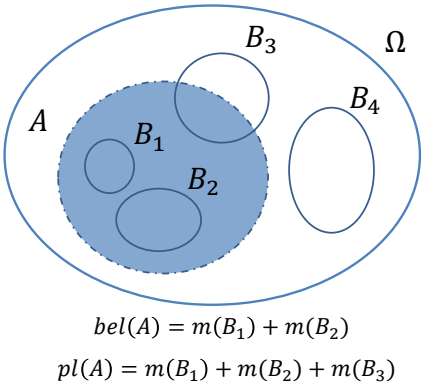

Figure 1.4 – Exemple Graphique des fonctions Bel et Pl

- **Affaiblissement :** L'intérêt de l'affaiblissement introduit dans [Shafer, 1976] est de maîtriser l'influence des sources selon leurs confiances avant de les combiner. Soit une fonction de croyance m fournie par une source S et un coefficient α avec $\alpha \in [0,1]$ qui représente le degré de confiance accordé à la source S avec :
  - ◇ $\alpha = 1$: La source S est totalement fiable.
  - ◇ $\alpha = 0$: La source S n'est pas du tout fiable.

  Un affaiblissement de la fonction de masse m par le facteur (1-α) produit une nouvelle fonction de masse m_α définie comme suit :

$$\begin{cases} m_\alpha(A) = \alpha.m_A, \quad \forall A \subseteq \Omega, A \neq \Omega \\ m_\alpha(\Omega) = 1 - \alpha + \alpha.m(\Omega). \end{cases} \tag{1.25}$$

  Dans la pratique le coefficient d'affaiblissement est souvent difficile à estimer.

Estimation : Toute la difficulté de la théorie des fonctions de croyance réside dans l'estimation des fonctions de masse. Plusieurs fonctions de masse ont été proposées dans la littérature et leurs choix relève de la modélisation et doit être fait selon les données et l'application recherchée.

- **Fonction à support simple :** Dans [Shafer, 1976] une première fonction de masse simple a été proposée, elle consiste à affecter toute la masse d'une source S_j à un sous-ensemble non vide A de 2^Ω. Ainsi, nous avons :

$$\begin{cases} m_j(A) = s \;\; s \in [0,1] \\ m_j(\Omega) = 1 - s \\ m_j(B) = 0 \;\; \forall B \in 2^\Omega, B \neq A, B \neq \Omega \end{cases} \tag{1.26}$$

  Si $s = 0$ alors $m_j(\Omega) = 1$ ce qui représente l'ignorance totale.

  Exemple 1.4. *Soit* $m_1(\{\text{Ouvert}\}) = 0.3$. *On aura alors* $m_1(\{\text{Ouvert}, \text{Fermé}\}) = 0.7$ *et* $m_1(\{\text{Fermé}\}) = 0$.

- **Fonction de croyance complémentaire :** Yager [Yager, 1987] propose une fonction de masse définie sur un sous ensemble A de Ω, pour une source S_j par :

$$\begin{cases} m_j(A) = s \;\; s \in [0,1] \\ m_j(A^c) = 1 - s \\ m_j(B) = 0 \;\; \forall B \in 2^\Omega, B \neq A, B \neq A^c \end{cases} \tag{1.27}$$

  Exemple 1.5. *Soit* $m_2(\{\text{Ouvert}\}) = 0.2$. *On aura alors* $m_2(\{\text{Fermé}\}) = 0.7$ *et* $m_2(\{\text{Ouvert}, \text{Fermé}\}) = 0$.

  Les fonctions de croyance complémentaires correspondent donc à la situation où toute la connaissance porte sur l'événement A et sa probabilité de réalisation est s.

- **Fonction de croyance sur les singletons :** Dans le cas où la source S_j n'a d'opinion que sur les singletons, la fonction de masse peut être définie comme :

$$\begin{cases} m_j(\omega_i) = M_i^j \\ m_j(A) = 0, si A \notin \{\omega_1\}, \{\omega_2\}, ..., \{\omega_n\} \end{cases} \tag{1.28}$$

  Exemple 1.6. *Soit* $m_3(\{\text{Ouvert}\}) = 0.2$. *On aura alors* $m_3(\{\text{Fermé}\}) = 0.8$ *et* $m_3(\{\text{Ouvert}, \text{Fermé}\}) = 0$.

D'autres approches ont été proposées pour estimer les masses de croyance ; voir notamment [Vannoorenberghe, 2003] et [Martin, 2005]. On peut citer entre autres,

la méthode basée sur les densités de probabilités [Appriou, 1991], [Smets, 1993], la méthode utilisant des arbres de décisions [Vannoorenberghe and Denoeux, 2002] et celle utilisant les k-plus proches voisins [Denoeux, 1995].

Combinaison : La combinaison des fonctions de masse issues des différentes sources, fondamentale pour la fusion de données, peut être réalisée suivant plusieurs opérateurs. On peut principalement distinguer deux types de combinaison : la *combinaison conjonctive* et la *combinaison disjonctive* d'où sont issus un grand nombre d'opérateurs, dont les opérateurs de combinaisons mixtes [Martin, 2005].

Soient S_1 et S_2 deux sources d'informations, et soient m_1 et m_2 les fonctions de croyance modélisant les informations données par ces deux sources respectivement :

- *La combinaison conjonctive* suppose que les deux sources sont totalement fiables et indépendantes entre elles. Elle consiste en une somme conjonctive des deux fonctions de croyance m_1 et m_2 relatives à une proposition A :

$$m_{\cap}(A) = \sum_{B \cap C = A} m_1(B) m_2(C) \quad \forall A \subseteq \Omega \tag{1.29}$$

  Cette règle de combinaison est associative, commutative, et possède un élément neutre.

  Exemple 1.7. *Soit un système de perception qui doit décider sur l'état ω de la porte exposée dans l'exemple 1.1. pour cela ce système utilise deux capteurs indépendants :*

  - ◇ *le capteur 1 donne l'information suivante :*
    $m_1(\{\text{Ouvert}\}) = 0.6$, $m_1(\{\text{Ouvert}, \text{Fermé}\}) = 0.1$, *et* $m_1(\{\text{Fermé}\}) = 0.3$.
  - ◇ *le capteur 2 donne l'information suivante :*
    $m_2(\{\text{Fermé}\}) = 0.5$, $m_2(\{\text{Ouvert}, \text{Fermé}\}) = 0.5$.

  Appliquant la règle de combinaison conjonctive sur cet exemple, on aura les résultats suivants.

  - ◇ $m_{\cap}(\{\text{Ouvert}\}) = m_1(\{\text{Ouvert}\})\ X\ m_2(\{\text{Ouvert}, \text{Fermé}\}) = 0.6\ X\ 0.5 = 0.3$.
  - ◇ $m_{\cap}(\{\text{Fermé}\}) = m_1(\{\text{Ouvert}, \text{Fermé}\})\ X\ m_2(\{\text{Fermé}\}) + m_1(\{\text{Fermé}\})\ X\ m_2(\{\text{Fermé}\}) + m_1(\{\text{Fermé}\})\ X\ m_2(\{\text{Ouvert}, \text{Fermé}\}) = 0.05 + 0.15 + 0.15 = 0.35$.

- $m_{\cap}(\{\text{Ouvert}, \text{Fermé}\}) = m_1(\{\text{Ouvert}, \text{Fermé}\})\ X\ m_2(\{\text{Ouvert}, \text{Fermé}\}) =$ *0.1 X 0.5 = 0.05.*
- $m_{\cap}(\emptyset) = m_1(\{\text{Ouvert}\})\ X\ m_2(\{\text{Fermé}\})$ *= 0.6 X 0.5 = 0.3.*

- *La combinaison disjonctive* [Smets, 1993] suppose que les deux sources d'informations sont indépendantes entre elles, et qu'au moins une des deux est fiable. La combinaison disjonctive de deux masses de croyance m_1 et m_2 relative à une proposition A notée $m_{\cup}(A)$ est donnée par :

$$m_{\cup}(A) = \sum_{B \cup C = A} m_1(B) m_2(C) \tag{1.30}$$

Cette règle de combinaison est associative, commutative, et ne possède pas d'élément neutre. Si m_1 et m_2 sont normalisées alors la fonction de masse résultante de la combinaison disjonctive est également normalisée.

Exemple 1.8. *Appliquant la règle de combinaison disjonctive sur les masses de croyances m_1, m_2 de l'exemple 1.7 on aura les résultats suivants :*

- $m_{\cup}(\{\text{Fermé}\}) = m_1(\{\text{Fermé}\})\ X\ m_2(\{\text{Fermé}\})$ *= 0.3 X 0.5 = 0.15.*
- $m_{\cup}(\{\text{Ouvert}, \text{Fermé}\})$ *=* $m_1(\{\text{Ouvert}, \text{Fermé}\}\ X\ m_2(\{\text{Ouvert}, \text{Fermé}\}) + m_1(\{\text{Ouvert}\})\ X\ m_2(\{\text{Fermé}\}) + m_1(\{\text{Ouvert}\})\ X\ m_2(\{\text{Ouvert}, \text{Fermé}\}) + m_1(\{\text{Fermé}\})\ X\ m_2(\{\text{Ouvert}, \text{Fermé}\}) + m_1(\{\text{Ouvert}, \text{Fermé}\}\ X\ m_2(\{\text{Fermé}\})$ *= 0.05 + 0.3 + 0.0.3 + 0.15 + 0.05 = 0.85.*
- $m_{\cup}(\{\text{Ouvert}\}) = 0.$
- $m_{\cup}(\emptyset) = 0.$

On a bien $m_{\cup}(\{\text{Ouvert}\}) + m_{\cup}(\{\text{Fermé}\}) + m_{\cup}(\{\text{Ouvert}, \text{Fermé}\}) = 0 + 0.15 + 0.85 = 1.$

- *Les opérateurs de gestion de conflit :* Combiner des informations issues de plusieurs sources qui ne sont pas parfaites peut faire apparaître un conflit entre ces sources. Comme le remarque Appriou [Appriou, 2002], ce conflit peut provenir d'un problème de modélisation, d'un manque d'exhaustivité des sources, d'un manque de fiabilité de celles-ci, ou encore du fait que ces sources n'observent pas le même phénomène. Dans ce dernier cas il faut bien sûr éviter de fusionner les sources d'information. La théorie des fonctions

de croyance offre un formalisme adapté à la fusion d'informations, dans lequel la considération du conflit est centrale. Pour cela plusieurs opérateurs [Lefevre et al., 2002][Lefevre et al., 2000] ont été définis afin de :

- ◇ analyser le conflit par rapport à la modélisation du problème.
- ◇ obtenir des masses normalisées (telles que $m(\emptyset) = 0$).

Le but de la gestion de conflit est de redistribuer la masse conflictuelle sur un ensemble de propositions. Ainsi la masse totale résultante de la fusion pour une proposition A est la somme des deux masses et s'écrit :

$$\begin{cases} m(A) = m_{\cap}(A) + m^c(A) \quad \forall A \in \Omega, A \neq \emptyset \\ m_{\cap}(\emptyset) = 0 \end{cases} \tag{1.31}$$

Le premier terme $m_{\cap}(A)$ correspond à la règle de combinaison conjonctive, le second noté $m^c(A)$ correspond à la masse de conflit affectée à la proposition A.

- ◇ *Règle de Dempster :* La règle de combinaison conjonctive produit une fonction de croyance non normalisée qui peut éventuellement être non nulle (c'est-à-dire pour laquelle $m_{\cap}(\emptyset) \neq 0$) représentant le conflit entre m_1 et m_2. La règle de Dempster vise à normaliser le résultat de l'opération conjonctive en divisant chaque masse par la constante de normalisation nommée K donnée par :

$$K = 1 - m_{\cap}(\emptyset) = \sum_{B \cap C = \emptyset} m_1(B) m_2(C) \tag{1.32}$$

Ainsi la combinaison de Dempster, appelée aussi somme orthogonale $\oplus$, s'écrit :

$$\begin{cases} m_{\oplus}(A) = \frac{1}{K}.m_{\cap}(A) = \frac{1}{K}.\sum_{B \cap C = A} m_1(B) m_2(C) \quad \forall \text{ A} \subseteq \Omega \\ m_{\oplus}(\emptyset) = 0 \end{cases} \tag{1.33}$$

Exemple 1.9. *Appliquant la règle de Dempster sur les masses de croyances m_1, m_2 de l'exemple 1.7 on aura les résultats suivants :*

- ✓ $m_{\cap}(\emptyset)$ =$m_1(\{\text{Ouvert}\})$ *X* $m_2(\{\text{Fermé}\})$ = *0.6 X 0.5 = 0.3.*
- ✓ $K = 1 - m_{\cap}(\emptyset)$ = *1 - 0.3 = 0.7.*
- ✓ $m_{\oplus}(\{\text{Ouvert}\}) = \frac{1}{K} m_{\cap}(\{\text{Ouvert}\}) = \frac{1}{0.7}$ *X 0.3= 0.4286.*
- ✓ $m_{\oplus}(\{\text{Fermé}\}) = \frac{1}{K} m_{\cap}(\{\text{Fermé}\}) = \frac{1}{0.7}$ *X 0.35 = 0.5.*

- ✓ $m_{\oplus}(\{\text{Ouvert}, \text{Fermé}\}) = \frac{1}{K} m_{\cap}(\{\text{Ouvert}, \text{Fermé}\}) \ \frac{1}{0.7} \ X \ 0.05 = 0.0714.$
- ✓ $m_{\oplus}(\emptyset) = 0.$

◇ *Opérateur de Yager :* La méthode proposée par Yager [Yager, 1987] suit le principe qu'au moins l'une des sources concernées par la fusion est fiable, mais sans savoir laquelle. Yager stipule donc que lorsque ces sources se contredisent alors la masse conflictuelle doit être placée sur Ω. Donc la masse m_Y résultante de la combinaison de deux sources d'information S_1 et S_2 est donnée par l'équation suivante :

$$\begin{cases} m_Y(A) = m_{\cap}(A) \ \ \forall A \neq \Omega, A \neq \emptyset \\ m_Y(\Omega) = m_{\cap}(\Omega) + m_{\cap}(\emptyset) \\ m_Y(\emptyset) = 0 \end{cases} \tag{1.34}$$

Exemple 1.10. *Appliquant la règle de combinaison de Yager sur les masses de croyances m_1, m_2 de l'exemple 1.7 on aura les résultats suivants :*

- ✓ $m_Y(\{\text{Ouvert}\}) = m_{\cap}(\{\text{Ouvert}\}) = 0.3.$
- ✓ $m_Y(\{\text{Fermé}\}) = m_{\cap}(\{\text{Fermé}\}) = 0.35.$
- ✓ $m_Y(\{\text{Ouvert}, \text{Fermé}\}) = m_{\cap}(\{\text{Ouvert}, \text{Fermé}\}) + m_{\cap}(\emptyset) = 0.05 + 0.3 = 0.35.$
- ✓ $m_Y(\emptyset) = 0.$

Cette règle de combinaison est commutative. Malheureusement, elle n'est pas associative. Donc elle est dépendante de l'ordre dans lequel on effectue la fusion. La règle de combinaison de Yager considère des sources d'informations non fiables, puisqu'elle considère le résultat de la fusion non fiable en cas de conflit. Cette méthode a pour effet de séparer la totalité de la masse conflictuelle, et donc de la faire intervenir plus dans le cadre de discernement Ω. Elle transfère la masse conflictuelle $m_{\cap}(\emptyset)$ sur la masse qui représente l'ignorance $m_{\cap}(\Omega)$, sans la répartir entre toutes les masses comme dans la règle de Dempster.

◇ *Opérateur de Dubois et Prade :* L'opérateur proposé par Dubois et Prade[Dubois and Prade, 1988b] repose sur l'hypothèse qu'au moins une des sources est fiable. Dans cet opérateur si le conflit est à l'origine de deux sous ensembles B et C dont l'union est égale à A alors cette masse de conflit est attribuée à la proposition A comme illustré dans l'équation

suivante :

$$\begin{cases} m_{DP}(A) = m_{\cap}(A) + \sum_{\{B \cup C = A, B \cap C = \emptyset\}} m_1(B) m_2(C) \quad \forall \text{ A} \neq \emptyset \\ m_{DP}(\emptyset) = 0 \end{cases} \tag{1.35}$$

Exemple 1.11. *Appliquant la règle de combinaison de Dubois et Prade sur les masses de croyances m_1, m_2 de l'exemple 1.7 on aura les résultats suivants :*

- ✓ $m_{DP}(\{\text{Ouvert}\}) = m_{\cap}(\{\text{Ouvert}\}) = 0.3.$
- ✓ $m_{DP}(\{\text{Fermé}\}) = m_{\cap}(\{\text{Fermé}\}) = 0.35.$
- ✓ $m_{DP}(\{\text{Ouvert, Fermé}\}) = m_{\cap}(\{\text{Ouvert, Fermé}\}) + m_1(\{\text{Ouvert}\})\ X\ m_2(\{\text{Fermé}\}) = 0.05 + 0.6\ X\ 0.5 = 0.35.$
- ✓ $m_{DP}(\emptyset) = 0.$

Cette règle de combinaison est plus précise dans la redistribution de la masse conflictuelle et donc plus informative que la règle proposée par Yager. On peut remarquer que cette combinaison est adaptative, elle utilise une approche conjonctive lorsque les sources sont d'accord, et une approche disjonctive en cas de conflit. L'opérateur de fusion proposé par Dubois et Prade est commutatif mais n'est pas associatif. Donc une stratégie permettant de combiner les sources dans un certain ordre doit être définie.

◇ *Opérateur de Smets :* Smets [Smets, 1990] considère que les sources à fusionner sont fiables. Partant de ce postulat, le conflit ne peut alors provenir que d'un problème mal posé : la non prise en compte d'une ou plusieurs hypothèses dans le cadre de discernement. Pour cela Smets recommande de ne pas redistribuer la masse conflictuelle sur l'ensemble des hypothèses possibles mais uniquement sur l'ensemble vide $\emptyset$, ceci afin de constater les problèmes de non exhaustivité de ce référentiel. C'est la notion de *monde ouvert* proposé par Smets [Smets and Kennes, 1994].

La combinaison de Smets est donc identique à la combinaison conjonctive :

$$\begin{cases} m_S(A) = m_{\cap}(A) \quad \forall \text{ A} \neq \emptyset \\ m_S(\emptyset) = m_{\cap}(\emptyset) \end{cases} \tag{1.36}$$

Exemple 1.12. *Appliquant la règle de combinaison de Smets sur les masses de croyances m_1, m_2 de l'exemple 1.7 on aura les résultats*

suivants :

- ✓ $m_S(\{\text{Ouvert}\}) = m_\cap(\{\text{Ouvert}\}) = 0.3.$
- ✓ $m_S(\{\text{Fermé}\}) = m_\cap(\{\text{Fermé}\}) = 0.35.$
- ✓ $m_S(\{\text{Ouvert}, \text{Fermé}\}) = m_\cap(\{\text{Ouvert}, \text{Fermé}\}) = 0.05.$
- ✓ $m_S(\emptyset) = m_\cap(\emptyset) = 0.3.$

L'ensemble vide $\emptyset$ peut être interprété comme une classe de rejet. Cette combinaison a les mêmes propriétés (associativité et commutativité) que la règle de Dempster.

Pour résumer, le tableau 1.1 présente ces règles, leurs hypothèses ainsi que leurs méthodes de distribution de $m(\emptyset)$ [Chebbah et al., 2011].

Règle de combinaison	**Hypothèse**	**Méthode de redistribution de $m(\emptyset)$**
Règle conjonctive de combinaison de Smets	Hypothèse du monde ouvert	$m(\emptyset)$ n'est pas redistribuée
Règle de Dempster	Hypothèse du monde fermé	$m(\emptyset)$ est redistribuée proportionnellement sur les éléments focaux
Règle de Yager	Hypothèse du monde fermé	$m(\emptyset)$ est affectée à Ω
Règle de Dubois et Prade	Hypothèse du monde fermé	La masse résultante de la combinaison des éléments focaux conflictuels est attribuée à l'union de ces éléments

Tableau 1.1 – Opérateurs de Gestion de conflit (Redistribution de $m(\emptyset)$)

Décision : Une fois que tous les éléments d'information ont été recueillis et toutes les croyances modélisées, mises à jour et combinées, le système doit prendre une décision sur la valeur de ω. Le but est donc de déterminer la solution la plus vraisemblable ω^* dans Ω, selon une règle donnée. Trois critères de décision se retrouvent couramment dans la littérature : le maximum de crédibilité, le maximum de plausibilité, et le maximum de probabilité pignistique.

- *Maximum de plausibilité :* Un premier critère est de choisir la solution ω^* la plus plausible (donnant le maximum de plausibilité), ainsi pour l'observation ω nous décidons ω^* si :

$$\omega^* = \underset{\omega_k \in \Omega}{argmax} Pl(\omega_k) \tag{1.37}$$

Ce critère permet de déterminer la meilleure hypothèse au sens du degré de confiance qu'on lui accorde et constitue en cela, un critère de décision *optimiste*.

Exemple 1.13. *la combinaison des masses de croyances m_1 et m_2 de l'exemple 1.7 par l'opérateur de Dempster fournit les fonctions de plausibilités suivantes :*

- $\diamond$ $Pl(\{\text{Ouvert}\}) = m_{\oplus}(\{\text{Ouvert}\}) + m_{\oplus}(\{\text{Ouvert}, \text{Fermé}\}) = 0.4286 + 0.0714 = 0.5$.
- $\diamond$ $Pl(\{\text{Fermé}\}) = m_{\oplus}(\{\text{Fermé}\}) + m_{\oplus}(\{\text{Ouvert}, \text{Fermé}\}) = 0.5 + 0.0714 = 0.5714$.

La décision sur la valeur de ω prise par le système selon le critère de maximum de plausibilité est $\omega^* = \{\text{Fermé}\}$.

- *Maximum de crédibilité :* Ce critère consiste a choisir la solution ω^* la plus crédible (dont la masse de croyance est maximum), c'est à dire choisir ω^* pour l'observation ω si :

$$\omega^* = \underset{\omega_k \in \Omega}{argmax} Bel(\omega_k) \tag{1.38}$$

Ce critère permet de déterminer la meilleure hypothèse en accordant un degré de croyance minimum à chacune d'entre elles. Il constitue en cela, un critère de décision *pessimiste*.

Exemple 1.14. *la combinaison des masses de croyances m_1 et m_2 de l'exemple 1.7 par l'opérateur de Dempster fournit les fonctions de crédibilités suivantes :*

- $\diamond$ $Bel(\{\text{Ouvert}\}) = m_{\oplus}(\{\text{Ouvert}\}) = 0.4286$.
- $\diamond$ $Bel(\{\text{Fermé}\}) = m_{\oplus}(\{\text{Fermé}\}) = 0.5$.

La décision sur la valeur de ω prise par le système selon le maximum de crédibilité est $\omega^* = \{\text{Fermé}\}$.

- *Maximum de probabilité pignistique :* dans le modèle des croyances transférables [Smets, 2000] [Smets, 2005] Smets propose un compromis aux critères du maximum de croyance et du maximum de plausibilité en introduisant la probabilité pignistique notée ($BetP$). Cette mesure approxime le couple (crédibilité, plausibilité) en equirépartissant la masse placée dans chaque sous ensemble sur les hypothèses qui le composent.

$$BetP(A) = \sum_{B \subseteq \Omega} \frac{|B \cap A|}{|B|} \frac{m(B)}{(1 - m(\emptyset))} \tag{1.39}$$

Où | B | est le cardinal de B. Ainsi le critère de maximum de probabilité pignistique revient à décider ω^* pour l'observation ω si :

$$\omega^* = \underset{\omega_k \in \Omega}{argmax} BetP(\omega_k) \tag{1.40}$$

Ce critère est un critère prudent car il permet de mettre en évidence l'impossibilité de décider entre plusieurs hypothèses de cadre de discernement, donc il est bien adapté dans le cas d'ambiguïtés. Cependant la probabilité pignistique nous fait repasser dans un contexte probabiliste. Les avis sont partagés sur ce point : certains, comme Smets, considèrent que ce passage est nécessaire pour la prise de décision, d'autres non.

Exemple 1.15. *La transformée pignistique des résultats de combinaison de Dempster sur les masses m_1 et m_2 de l'exemple 1.7 fournit les distributions pignistiques suivantes :*

- $BetP(\{\text{Ouvert}\}) = \frac{2m_{\oplus}(\{\text{Ouvert}\})+m_{\oplus}(\{\text{Ouvert, Fermé}\}}{2} = 0.4643.$
- $BetP(\{\text{Fermé}\}) = \frac{2m_{\oplus}(\{\text{Fermé}\})+m_{\oplus}(\{\text{Ouvert, Fermé}\}}{2} = 0.5357.$

La décision sur la valeur de ω prise par le système selon le maximum de probabilités pignistique est $\omega^* = \{\text{Fermé}\}$.

1.3 Tolérance aux fautes en fusion de données

A notre connaissance, peu de travaux existent sur la tolérance aux fautes dans la fusion de données. Les approches que nous avons trouvées dans la littérature utilisent principalement la tolérance aux fautes par duplication/comparaison pour tolérer des fautes physiques. Nous avons classé ces approches en deux catégories : La duplication basée sur un modèle analytique, et la duplication basée sur la redondance matérielle.

1.3.1 Robustesse et tolérance aux fautes

Le concept de robustesse est largement utilisé dans la communauté robotique, mais il n'existe pas de définition précise de ce terme utilisée dans cette communauté. Un système est considéré robuste s'il est capable de fournir un service satisfaisant en dépit de situations difficiles et hasardeuses. Une telle robustesse peut être vue comme une réponse à la variabilité du contexte d'exécution auquel est confronté le système. Cette variabilité est principalement due à l'environnement ouvert dans lequel le système fonctionne, mais également à des incertitudes et imprécisions résultant de

l'incapacité du système à observer correctement son environnement et son propre état, et de la difficulté de prévoir les conséquences de ses propres actions.

La notion de robustesse dans les domaines de la robotique et de la perception est proche de celle de la tolérance aux fautes en sûreté de fonctionnement : les deux visent à accroître la capacité du système à fournir un comportement correct, mais la tolérance aux fautes vise un sous-ensemble spécifique des situations rencontrées par un système robotique : les fautes. Dans notre étude, nous allons distinguer les deux concepts par l'origine des situations adverses : soit interne soit externe au système de perception. Ainsi, nous conservons les deux définitions suivantes [Lussier, 2007] :

- *La robustesse* est la délivrance d'un service correct dans des situations implicitement adverses dues à l'incertitude de l'environnement du système (par exemple un changement dans les conditions d'éclairage affectant les caméras lors de la conduite, ou de faux positifs détectés à l'aide d'un laser, sous la pluie qui enregistre les gouttes comme des obstacles).

- *La tolérance aux fautes* est la délivrance d'un service correct en dépit de fautes affectant les ressources du système (un dysfonctionnement du capteur ou une faute logicielle affectant les algorithmes de fusion de données) ; par exemple une caméra bloquée sur la dernière image acquise est une faute interne au capteur par conséquent interne au systeme de perception dans son ensemble.

Il est à noter que cette définition de la robustesse est similaire à celle utilisée en sûreté de fonctionnement, contre les *fautes externes*. Dans la pratique toutefois, il n'est pas toujours facile de distinguer les situations dues à un environnement incertain des situations dues à des fautes affectant les ressources du système.

1.3.2 Méthodes de tolérance aux fautes en fusion de données

La tolérance aux fautes apparaît dans de nombreuses études de fusion de données. Les différentes approches proposées dans la littérature utilisent principalement la tolérance aux fautes par *duplication / comparaison*. On classe ces approches en deux classes : celles basées sur un modèle analytique, et celles basées sur la redondance matérielle. Notons que ces classes ne sont pas mutuellement exclusives, en effet les deux utilisent la redondance matérielle des sources de données, cependant la première ajoute un modèle analytique du système comme étant une diversification de capteur afin d'augmenter le niveau de redondance, et généralement le type de fusion de données appliqué dans cette première catégorie est le filtre de Kalman. Ces approches sont détaillées ci dessous.

1.3.2.1 Duplication basée sur un modèle analytique

Dans ces approches un modèle analytique du système est utilisé comme une diversification de capteurs physiques afin d'exploiter une telle redondance pour détecter des erreurs sur les sorties des capteurs. Généralement le mécanisme de fusion de données appliqué dans ce type d'approches est le filtre de Kalman. En effet le filtre de Kalman utilise le modèle du système pour estimer une mesure redondante à celle délivrée par un capteur réel. Ces approches emploient le résidu issu de la combinaison de la mesure estimée par le modèle du système et celle délivrée par le capteur comme un indicateur d'erreur.

Dans [Escamilla-Ambrosio and Mort, 2001], une architecture hybride de fusion de données tolérante aux fautes intégrant les techniques de filtre de Kalman et de la logique floue est proposée. Deux niveau de fusion sont définis : dans le premier niveau n filtres de Kalman sont exécutés en parallèles utilisant n sorties de capteurs redondants. A ce niveau les fautes transitoires des capteurs sont détectées à l'aide d'une technique de résidus. En effet, l'évaluation du résidu de chaque filtre permet de détecter une faute transitoire lorsque la valeur d'un filtre change brutalement. Dans le second niveau de fusion une technique de vote [Caspi and Salem, 2000] est réalisée sur les sorties de chaque filtre : si un des filtres estimés diffère nettement des autres cela signifie qu'une faute persistante sur le capteur correspondant est présente. Pour des fins de tolérance aux fautes sa valeur ne sera plus utilisée dans le second niveau de fusion qui consiste à faire une somme pondérée des estimations issues des n filtres implémentés.

Dans [Zug and Kaiser, 2009] les auteurs présentent le concept d'un capteur virtuel [2] tolérant aux fautes en combinant des estimations issues d'un modèle analytique du système et les données des capteurs redondants. En effet ce capteur virtuel combine plusieurs capteurs abstraits [3] au moyen de la fusion de données. Dans cette approche deux niveaux de fusion sont définis : une fusion locale dans chaque capteur abstrait, et une fusion globale dans tout le capteur virtuel. Au niveau local, les données brutes issues de chaque capteur réel sont fusionnées avec les données de position provenant d'un modèle mathématique du système et avec les résultats de l'estimation de la position globale précédente issue du capteur virtuel par le biais du filtre de Kalman. La détection de fautes est réalisée au niveau de chaque capteur abstrait : si la mesure fournie par le capteur réel diverge de celle fournie par le modèle mathématique du système, une erreur matérielle est détectée. Par la

[2] Virtual sensor : A basic virtual sensor combines multiple singleton sensors of the same type and provides a better reliability.

[3] abstract sensor : one that consumes the output of a physical transformation process (concrete sensor) and calculates a validity interval.

suite une fusion globale par moyenne pondérée des sorties des capteurs abstraits est réalisée au niveau du capteur virtuel pour obtenir une estimation de position plus fiable du robot. Les auteurs ont validé leur approche à travers un exemple de simulation de robot utilisant trois types de capteurs : un capteur ultrason, une caméra, et un laser. Ces capteurs délivrent la position du robot à chaque instant, cette dernière étant évaluée par un modèle mathématique au niveau des capteurs abstraits pour une détection éventuelle des erreurs matérielles. Par la suite une fusion globale est réalisée pour avoir une localisation plus précise du robot.

Dans [Morales et al., 2008] les auteurs proposent une technique de localisation 2D d'un robot mobile en utilisant la fusion de données par filtrage de Kalman. Trois capteurs sont employés : une centrale inertielle, les encodeurs du robot, ainsi qu'un GPS. La détection d'erreurs consiste à rejeter les données GPS si elles sont inconsistantes avec l'état prédit par le filtre de Kalman. Une telle détection se fait par le test statistique NIS (Normalized Innovation Squared) [Lancaster and Seneta, 1969] sur le résidu du filtre. Une validation sur une application réelle de localisation est réalisée pour montrer l'efficacité de l'approche proposée.

[Roumeliotis et al., 1998] proposent une approche de détection, d'isolation et de re-configuration de fautes (FDIR : fault detection, isolation and re-configuration) dans le cadre d'une application de navigation de véhicule terrestre autonome (AGV). Dix capteurs sont fusionnés au moyen de filtres de Kalman pour obtenir plusieurs estimations de la position d'un robot mobile. Les mesures des capteurs et les données fusionnées sont ensuite combinées en un ensemble d'équations de parité linéairement indépendantes, ce qui conduit à la génération d'une banque de résidus. Une faute dans l'un des dix capteurs provoque la croissance d'un sous-ensemble unique de ces résidus, ce qui permet la détection de la faute et son isolation. Un système de contrôle est mis en place pour assurer la re-configuration automatique en réalisant la localisation du robot avec le sous-ensemble de capteurs qui est considéré comme fonctionnel.

[Allerton and Jia, 2008] présentent les approches de fusion de données pour les systèmes de réseau inertiels. Dans ces systèmes, les auteurs utilisent des nœuds redondants structurés hiérarchiquement. La plupart des nœuds jouent le rôle d'esclaves tandis que l'un joue le rôle de maître, qui réalisera la fusion globale. Différentes matrices de transformation (rotation et translation) sont développées pour assurer l'alignement des mesures. Un filtre de Kalman est utilisé comme algorithme de fusion au niveau de chaque nœud esclave en utilisant un modèle du système et la mesure délivrée par l'IMU pour estimer l'état du système (Measurement fusion). Sous l'hypothèse que le résidu issu du filtre de Kalman suit une loi gaussienne de moyenne nulle en fonctionnement normal, les auteurs proposent

de réaliser des tests statistiques sur cette grandeur pour contrôler la divergence du filtre et la détection de fautes de capteurs. Par la suite le nœud maître réalise la fusion des différents états estimés par les nœuds esclaves pour avoir un état global plus précis (state fusion). A ce niveau la redondance des IMU (nœuds esclaves) est exploitée pour assurer le masquage de fautes. Une validation de l'approche proposée est réalisée par une simulation d'un réseau inertiel sur Matlab.

L'approche [Roumeliotis et al., 1998] utilise une banque de filtres de Kalman, chacun étant associé à une faute spécifique du capteur pour assurer la détection. L'isolation des fautes est ainsi obtenue par l'étude et l'analyse des résidus générés. L'approche est démontrée sur un robot d'entraînement différentiel équipé d'encodeurs des roues (odométrie) et d'un gyromètre.

1.3.2.2 Duplication basée sur la redondance matérielle

Les approches basées sur la redondance matérielle combinent plusieurs sources de données en utilisant la fusion de données. Ces approches contrairement aux approches basées sur le modèle analytique du système analysent certains paramètres internes aux mécanismes de fusion de données (tels que le conflit dans le cadre des fonctions de croyance) afin d'assurer la tolérance aux fautes.

Dans [Ricquebourg et al., 2007b], les auteurs proposent une méthode de détection d'un dysfonctionnement capteur par l'analyse temporelle du conflit résultant de la fusion, dans le cadre des croyances transférables (TBM) de Smets [Smets, 1998][Smets, 2000]. Les auteurs proposent de fusionner les sources de données par paires et tiennent compte de l'évolution des conflits (tels que définis par Smets) résultant de la fusion. Une source est considérée comme défectueuse si son conflit avec les autres est élevé. Après la reconnaissance de la source défectueuse les auteurs suggèrent d'affaiblir son influence sur la décision finale, en associant les indices de fiabilité et en utilisant la technique d'affaiblissement proposée par Shafer [Shafer, 1976]. Cela affaiblit les masses des sources et augmente leurs facteurs d'affaiblissement sans écarter les sources conflictuelles complètement. Les auteurs implémentent leur approche pour une application de perception de contexte pour l'habitat communicant et le maintien à domicile des personnes âgées en utilisant trois types de capteurs : un capteur de pression, une camera webcam, et un capteur de suivi de personnes. L'objectif de cette application est la détection des personnes assises sur une chaise dans une chambre. Les auteurs ont montré la faisabilité de leur méthode, en particulier par la mise en œuvre de deux expériences : une première dans laquelle il n'y a pas de conflit entre les sources ; un second cas où un conflit apparaît entre les sources combinées dû à une faute dans l'une des sources. Cette

analyse de conflit a permis d'isoler la source défaillante et de déterminer précisément la phase de dysfonctionnement, ce qui entraîne l'affaiblissement de la contribution de la source défaillante dans la décision issue de la fusion globale.

Dans [Delmotte and Gacquer, 2008], un algorithme pour détecter une source défectueuse par analyse de la fiabilité de chaque source est proposé. Pour chaque source sont associées deux fiabilités : la fiabilité statique représentant la qualité de la source alors que la fiabilité dynamique indique son conflit avec les autres. La fiabilité globale de la source est alors calculée à partir des deux fiabilités précédentes. Cette fiabilité globale sert comme un facteur d'affaiblissement qui pondère les masses de croyance avant qu'elles soient combinées. En supposant que le conflit provient d'une source défectueuse, les auteurs analysent le facteur d'affaiblissement (fiabilité globale des sources) pour détecter la source erronée en utilisant une méthode de seuillage. Une validation numérique de l'approche proposée est réalisée en utilisant des fonctions à support simple pour la modélisation des fonctions de croyance. Pour montrer l'effet de l'isolation de la source défaillante sur la décision finale de la fusion, les auteurs ont comparé leur résultat avec les résultats des différents opérateurs de combinaison existant dans la littérature. Dans [Delmotte, 2007], une approche similaire a été proposée par les mêmes auteurs, dans le contexte spécifique de la théorie des possibilités.

Dans [Shu-qing and Sheng-xiu, 2010] une technique adaptative a été utilisée pour pondérer les sorties des capteurs dans le processus de fusion de données multi-capteurs. Cette technique utilise l'écart type de chaque capteur (estimé par des statistiques et un facteur temporel lié aux données précédentes) pour calculer un facteur d'affaiblissement des sorties des capteurs et l'utiliser dans le processus de fusion de données pour assurer la tolérance aux fautes et affaiblir les effets négatifs du vieux matériel. Une validation par simulation de l'algorithme proposé est réalisé. Une simulation de comportement nominal est d'abord effectuée, et une autre avec un capteur fautif. Les auteurs ont démontré que leur algorithme capture bien le capteur erroné en temps voulu. Par la suite pour assurer sa tolérance un facteur de poids lui est attribué pour réduire son effet sur le résultat de la fusion.

Dans [Peng et al., 2012] Peng et al proposent une méthode de diagnostic dans les circuits analogiques qui combinent les réseaux de neurones et le raisonnement evidentiel. Cette méthode suit deux étapes. Tout d'abord, un pré-diagnostic est mis en œuvre en utilisant un réseau neuronal sur un ensemble présélectionné de signaux de test pour construire une correspondance entre la sortie des capteurs et des états erronés. Deuxièmement, un raisonnement evidentiel en utilisant la théorie Dempster-Shafer est utilisée pour fusionner les différents modèles de fautes possibles pour obtenir le bon. À notre avis, un tel diagnostic ne pourrait pas être utilisé sur

des systèmes critiques, parce que le mapping dans la première étape est basé sur un apprentissage, qui pourrait ne pas être exhaustif.

D'autres exemples similaires de tolérance aux fautes dans la fusion de données sont mentionnés dans [Zhuo-hua et al., 2005], un article qui examine les techniques de tolérance aux fautes dans les systèmes robotiques à roues.

1.3.3 Critiques des approches existantes

De notre point de vue, les approches proposées se concentrent sur la tolérance aux fautes matérielles liées aux sources de données fusionnés, en utilisant directement les mécanismes de fusion de données sans garantie que ces derniers soient exempts de fautes. Or, ces mécanismes déclaratifs sont justement complexes à développer et à valider. Cela est dû à leur comportement non prédictible qui les rend difficilement vérifiables par des approches formelles et à leur paradigme de programmation, encore dépendant de variables empiriques. De plus en robotique mobile ces approches sont utilisées pour la perception multi-capteurs dans des environnements ouverts ce qui rend leur validation par des tests une tâche coûteuse et longue. Ainsi il nous paraît difficile d'avoir une confiance justifiée dans leurs services.

La tolérance aux fautes grâce à la fusion de données peut seulement être faite à notre avis en essayant de tolérer d'abord les fautes logicielles dans le processus de fusion de données, ou de garantir par des méthodes formelles que ce processus est digne de confiance.

Dans la littérature nous avons trouvé seulement deux articles traitant ce problème. Bien que les auteurs utilisent la fusion de données pour la tolérance aux fautes matérielles dans les sources de données fusionnées, ils garantissent d'abord par une analyse formelle que le comportement de la fusion de données est correct, ce qui augmente la confiance que l'on peut avoir dans ces mécanismes de fusion de données :

- Dans [Uhlmann, 2003] une approche de fusion de données basée sur l'intersection des covariances CI(covariance intersection) et l'union des covariances CU(covariance Union) est proposée pour assurer la cohérence et la tolérance aux fautes pour la gestion des informations dans les réseaux distribués. La solution CI permet de s'assurer que le résultat de la fusion des estimations cohérentes est cohérent. La solution CU complète la CI en veillant à ce que le résultat de la fusion des estimations incohérentes reste toujours cohérent. Ce mécanisme est validé par une analyse formelle de ses propriétés, et tolère ainsi des fautes sur les capteurs à travers le masquage.

- Dans [Okello, 1993] une approche de fusion de données binaires dans un système distribué est proposée. Cette approche fournit la tolérance aux fautes en analysant formellement les niveaux de confiance associés aux capteurs. Cette analyse permet d'améliorer la robustesse de la fusion de données booléennes et d'éliminer ses faiblesses, afin de réduire ou même d'éliminer les fausses alarmes.

La solution que nous proposons dans le second chapitre se base sur une analyse formelle exhaustive du comportement du système en regard de certains paramètres de modélisation du problème, justifiant pour nous une plus grande sûreté de fonctionnement. Le prix de cette garantie est que l'analyse peut être complexe, voire irréalisable dans certains cas. Nous proposons également une architecture tolérante aux fautes prenant en compte les fautes logicielles dans les mécanismes de fusion de données.

1.4 Conclusion

Dans ce chapitre nous avons introduit la terminologie et les concepts de base des deux domaines traités dans ces travaux de thèse, à savoir la sûreté de fonctionnement informatique, et les systèmes de fusion de données multi-capteurs. Nous avons aussi présenté un état de l'art sur les techniques de tolérance aux fautes en fusion de données. Cette étude nous a permis d'établir les conclusions suivantes :

- La mise en œuvre de la tolérance aux fautes est concentrée sur les fautes matérielles. En outre, les techniques proposées dans la littérature détectent et tolèrent les fautes liés à des capteurs physiques en utilisant des processus de fusion de données qui sont difficiles à concevoir et à valider. Ces processus nous paraissent peu dignes de confiance, d'où la nécessité d'analyser formellement ces mécanismes de fusion afin de les rendre sûrs de fonctionnement (ceci est l'objectif de second chapitre), ou encore de s'assurer de leur bon fonctionnement par des mécanismes de tolérance aux fautes les ciblant spécifiquement.

- On trouve peu de techniques visant explicitement la tolérance aux fautes logicielles. Or, pour nous les fautes affectant les composants logiciels sont importantes à considérer.

Étant donné que la perception est une entrée fondamentale pour les systèmes robotiques, son comportement doit être toujours correct. La sûreté de fonctionnement doit cependant chercher à minimiser l'apparition de comportements inacceptables, voire dangereux de ces systèmes de perception. Dans la suite de ce manuscrit nous

proposons des méthodes de tolérance aux fautes qui permettent à ces systèmes d'avoir un comportement correct même en présence de fautes.

Mécanismes de tolérance aux fautes pour la fusion de données

Sommaire

2.1 Introduction

Après l'étude de l'état de l'art, nous avons identifié plusieurs pistes d'étude pour garantir la sûreté de fonctionnement des systèmes de perception multi-capteurs, en particulier leur tolérance aux fautes internes. Pour cet objectif nous proposons dans ce chapitre deux approches : la première utilise la méthode de tolérance aux fautes classique (*duplication et comparaison*) dans la fusion de données (Figure 2.1), et la seconde est basée sur la redondance utilisée dans certaines types de fusion de données (Figure 2.2). Dans ce second cas, nous étudions les services de tolérance aux fautes intrinsèquement offerts par la fusion de données, avec une analyse mathématique du mécanisme de fusion de données nécessaire pour garantir le bon comportement du système de perception.

Les deux architectures décrites dans ce chapitre implémentent deux algorithmes de fusion de données entre deux types de capteurs redondants et différents : l'architecture basée sur la fusion ajoute un troisième bloc de fusion globale combinant les sorties des deux blocs de fusion implémentés (Figure 2.2), tandis que l'architecture de duplication et comparaison ajoute un mécanisme de comparaison et

diagnostic (Figures 2.1). Dans les deux architectures nous garantissons des services de détection et de recouvrement adaptés aux systèmes de perception multi-capteurs pour assurer leur fiabilité. Nous concluons ce chapitre par une étude qualitative des deux approches proposées sur un cas d'étude très simple.

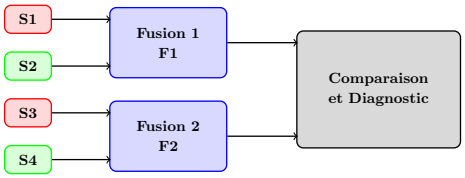

Figure 2.1 – Architecture de tolérance aux fautes par Duplication/Comparison

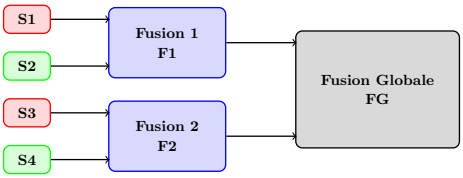

Figure 2.2 – Architecture de tolérance aux fautes par fusion de données

2.2 Tolérance aux fautes par Duplication - Comparaison

Dans cette section nous présentons l'architecture de détection et de rétablissement pour la perception multi-capteurs basée sur la méthode classique de *duplication / comparaison*. La Figure 2.3 détaille cette architecture générique. Elle implémente deux branches parallèles et indépendantes, chacune exécute un mécanismes de fusion de données utilisant des capteurs redondants ou diversifiés pour percevoir l'état du robot et de son environnement. Cette architecture peut tolérer une faute matérielle liée aux blocs de capteurs et détecter une faute logicielle liée aux blocs de fusion, sous l'hypothèse de la faute simple matérielle ou logicielle (pas plus d'une seule faute active au même instant dans le système). Ces services de tolérance aux fautes pourraient être étendus à des fautes multiples en augmentant le niveau de redondance matérielle et logicielle de l'architecture. Une telle redondance pourrait aussi être exploitée pour tolérer les fautes logicielles détectées en utilisant la diversification des mécanismes de fusion de données et la méthode de vote majoritaire.

L'architecture présentée dans la figure 2.3 compare les résultats des deux blocs de fusion afin de détecter un écart significatif entre les deux sorties de fusion, cet écart indiquant la présence d'une erreur. Les sorties des capteurs et les valeurs des résidus issus de la fusion sont utilisées pour déterminer si cette erreur est due à une faute matérielle dans les capteurs ou à une faute logicielle dans les algorithmes de fusion, afin d'identifier la sortie correcte et d'assurer le rétablissement.

Les détails de cette approche sont présentés dans les sous sections suivantes. Un

exemple d'implémentation sur un cas réel pour la localisation des véhicules sera présenté dans le chapitre 3.

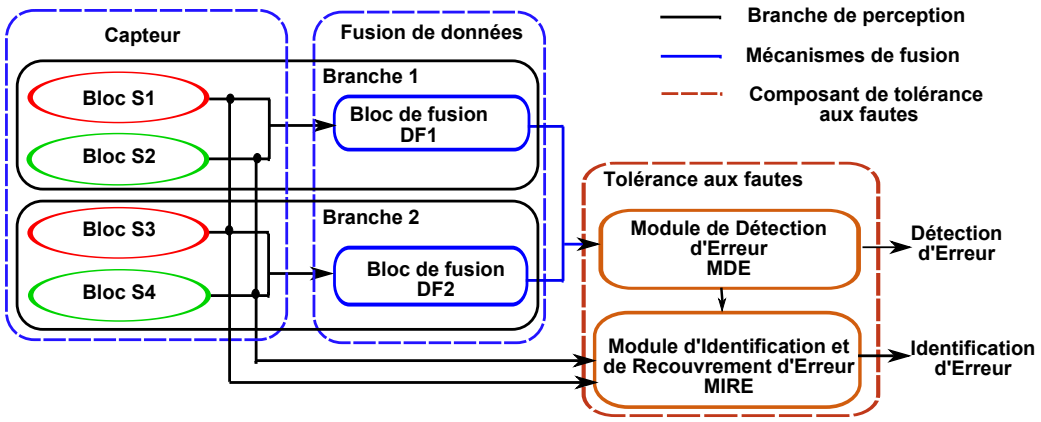

Figure 2.3 – Architecture de duplication-comparaison pour la tolérance aux fautes en perception multi-capteurs

2.2.1 Services de détection et de recouvrement de fautes

Notre architecture met en œuvre deux branches parallèles, chacune exécutant un bloc de fusion de données et utilisant deux blocs de capteurs : La première branche (S_1, S_2, DF_1) combine les sorties des blocs de capteurs S_1 et S_2 au sein du bloc de fusion DF_1, et la seconde branche (S_3, S_4, DF_2) combine les sorties des blocs de capteurs S_3 et S_4 au niveau du bloc de fusion DF_2 . Les sorties de chaque composant dupliqué sont comparées avec son redondant diversifié : par exemple, dans la Figure 2.3 la sortie de DF_1 est comparée à la sortie de DF_2, la sortie du bloc de capteur S_1 est comparée à la sortie du bloc S_3, et la sortie du bloc S_2 est comparée à la sortie du bloc S_4. Les sorties des blocs de fusion sont comparées pour détecter une erreur dans le système, et les sorties des blocs de capteurs sont utilisées pour diagnostiquer l'erreur détectée et déterminer la sortie correcte du système de perception. Notre architecture est ainsi constituée de deux modules :

1. *Le Module de Détection d'Erreur(MDE)* détecte une erreur possible dans le système en comparant les sorties des deux blocs de fusion de données. Cette comparaison consiste à calculer une distance Δ_{DF_1,DF_2} entre les sorties des deux blocs de fusion de données, et à la comparer à un seuil spécifique $Thrs_{Det}$. Le

fonctionnement de ce module est résumé dans l'algorithme 2.1.

Algorithm 2.1: Algorithme de détection d'erreur

Data: DF_1,DF_2 : sortie de fusion
Result: E : indicateur d'Erreur

```
1 Calculer Δ_{DF1,DF2}
2 if Δ_{DF1,DF2} > Thrs_{Det} then
3 |   E ← {OUI} ;
  |   /* Comparer les sorties des blocs capteurs et les résidus
  |      pour le diagnostic et le rétablissement                  */
4 else
5 |   E ← {NON} ;
6 end
```

2. *Le Module d'Identification et de Recouvrement d'Erreur (MIRE)* diagnostique l'erreur détectée, et identifie la sortie correcte du système. Ce diagnostic se fait en deux étapes :

   - *La comparaison des sorties des capteurs (SC)* détermine si l'erreur détectée est matérielle ou logicielle. Ceci est réalisé en comparant les sorties de différents blocs capteurs (S_1 et S_3 d'une part, S_2 et S_4 d'autre part) :

     ◇ Si la sortie d'un bloc capteur s'écarte significativement de son dual alors le système diagnostique une erreur matérielle sur l'un de ces deux capteurs. Le capteur défectueux sera identifié dans la comparaison des résidus.

     ◇ Si les sorties des capteurs sont similaires, le système diagnostique une erreur logicielle. En absence de plus de redondance, nous ne pouvons rien conclure et recommandons de mettre le système erroné dans un état sûr. Cependant, un autre bloc fusion de données diversifié DF_3 utilisant deux sorties parmi S_1, S_2, S_3 et S_4 pourrait être mis en œuvre pour diagnostiquer le bloc de fusion de données défectueux et rétablir le système.

   - *La comparaison des résidus (RC)* identifie, dans le cas d'une erreur matérielle, le bloc de fusion de données défaillant en comparant les valeurs de résidus de chaque bloc de fusion de données. La valeur de résidu la plus élevée indique un conflit dans les sources combinées, et donc la branche erronée. Nous faisons confiance ici aux mécanismes de fusion car sous l'hypothèse de la faute unique nous avons vérifié dans l'étape

de comparaison des sorties des capteurs que la divergence entre les deux branches est due à une erreur de capteur et non aux mécanismes de fusion.

Connaissant maintenant le couple ((S_1 ; S_3) ou (S_2 ; S_4)) contenant le capteur défaillant et la branche erronée, le système a identifié le capteur erroné, et peut se rétablir en utilisant les valeurs de sortie de la branche sans erreur.

2.2.2 Analyse qualitative

L'architecture proposée fournit les services de tolérance aux fautes suivants :

- *Détection et rétablissement d'une faute matérielle* : Elle détecte et permet au système de se rétablir après une erreur matérielle liée aux blocs de capteurs (S_1, S_2, S_3 ou S_4).
- *Détection d'une faute logicielle* : Elle détecte une faute logicielle liée aux algorithmes de fusion de données (DF_1, DF_2). Une telle faute peut être tolérée par un troisième bloc de fusion diversifié et une méthode de vote.

Pour assurer ces services, notre architecture requiert une comparaison des sorties des blocs des capteurs :

- la comparaison peut être effectuée directement sur la sortie des capteurs ou sur des données pré-traitées. Par exemple dans notre étude de cas (Chapitre 3), la dérivée de la vitesse des roues arrières est calculée avant d'être comparée à l'accélération issue du système de navigation inertiel.
- la comparaison peut être faite entre deux blocs de capteurs, chacun englobant un ou plusieurs capteurs, pour lesquels toutes les sorties utilisées d'un bloc ont une contrepartie (brute ou pré-traitée) dans le second bloc.

Rappelons que nous travaillons sous l'hypothèse de faute unique (pas plus d'une faute active au même instant dans le système), mais que cette hypothèse pourrait être retirée en utilisant plus de redondance, à la fois matérielle et logicielle.

2.3 Tolérance aux fautes par fusion de données

Après avoir présenté la première approche de duplication/comparaison, nous analysons dans cette section la seconde approche basée sur la tolérance aux fautes intrinsèquement fournie par certains types de fusion de données. Une analyse formelle est nécessaire dans ce cas pour garantir la tolérance aux fautes. Un exemple simple est détaillé, basé sur un cas d'application repris de [Ricquebourg et al., 2007a], où

elle avait été proposée dans le cadre d'une architecture de perception de contexte pour l'habitat communicant et le maintien à domicile des personnes âgées.

2.3.1 Principe de l'approche

L'approche que nous proposons dans cette section utilise la redondance de la fusion de données, et emploie des capteurs compétitifs et complémentaires. Pour modéliser et combiner les informations issues de ces capteurs nous utilisons les fonctions de croyance, le cadre théorique englobant les autres théories (probabilités, possibilités). Dans cette approche aucun mécanisme de tolérance aux fautes n'est utilisé. En effet c'est une analyse formelle de la fusion, elle consiste en une validation mathématique de la fusion afin d'assurer son bon fonctionnement et offrir une confiance justifiée dans le service délivré. Étant donné que les mécanismes de fusion sont difficiles à valider formellement, et pour montrer notre approche nous nous sommes basés sur un cas théorique très simple afin de simplifier notre étude.

2.3.2 Présentation du cas d'étude

Le système mis en œuvre dans cette étude est un cas théorique simple considéré pour analyser les services de tolérance aux fautes pouvant être fournis par la fusion de données. Sa fonction consiste à détecter la présence d'une personne sur une chaise. Pour atteindre cet objectif deux capteurs sont utilisés : un capteur de pression placé sous la chaise, et une caméra installée à l'avant de la chaise pour détecter une silhouette comme présenté sur la Figure 2.4.

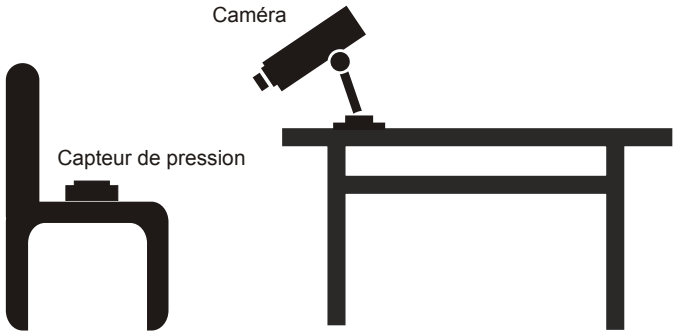

Figure 2.4 – Scénario considéré

Dans ce scénario, nous considérons que les capteurs non défaillants fournissent une sortie non bruitée et sans faux positif : la sortie de la caméra et du capteur de pression est 1 si une personne est effectivement sur la chaise, et 0 sinon. En d'autres termes, nous nous concentrons sur les fautes internes au système de perception plutôt

que sur les aléas environnementaux : l'étude se concentre sur la tolérance aux fautes plutôt que la robustesse.

2.3.3 Terminologies et notations

Nous introduisons dans la table 2.1 les différentes terminologies et notations utilisées dans la suite de ce chapitre.

Notation	**Description**
P_j C_j F_j Avec $j \in \{1,2\}$	Capteur de pression j Caméra j Bloc de fusion j
FG	Fusion Globale
$threshold$	Seuil de decision
p^i_{trust} $p_i(\exists)$ Avec $i \in \{\text{P,C}\}$	Facteur d'affaiblissement associé au capteur i Sortie du capteur i Avec P pour pression et C pour caméra

Tableau 2.1 – Notations pour l'étude de cas

2.3.4 Objectif de l'étude

Le but de cette étude est de déterminer les critères de détection et de rétablissement d'une erreur dans le système de perception, englobant à la fois les capteurs et les algorithmes de fusion. Les fautes visées sont les fautes matérielles liées aux capteurs (caméra et capteur de pression), et les fautes logicielles associées aux algorithmes de fusion. Cette architecture se compose de deux branches identiques comprenant une caméra, un capteur de pression, et un mécanisme de fusion entre ces deux capteurs (Figure 2.6). Contrairement à l'architecture de duplication/comparaison présentée dans la section précédente (2.2) cette architecture ajoute un algorithme de fusion globale combinant les sorties issues de chacune des branches comme indiqué dans la Figure 2.5.

Avec cette configuration, nous proposons trois services de tolérance aux fautes :

1. *Détection :* L'architecture peut détecter (mais pas diagnostiquer) une erreur dans les quatre capteurs, et éventuellement, dans l'algorithme de fusion de données.

2. *Recouvrement par compensation :* une fois une erreur détectée, la sortie correcte est déduite à partir de la branche sans erreur.

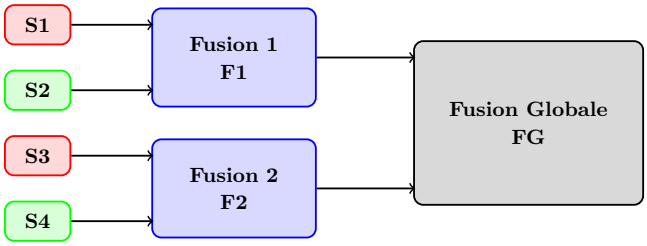

Figure 2.5 – Architecture du fusion de données

3. *Masquage :* L'architecture peut tolérer une erreur matérielle sans la détecter : en dépit d'un capteur erroné, le système donne le résultat correct.

Notons que le masquage de fautes peut être risqué si le système n'est pas entretenu régulièrement. En effet, comme une erreur de capteur est masquée et non détectée, le système fonctionnera sans symptômes externes jusqu'à ce qu'une autre faute capteur soit activée. Cette deuxième faute ne sera pas tolérée, comme le niveau de redondance dans notre architecture permet la tolérance d'une seule faute, et le système de perception peut défaillir. Notons cependant que plus de fautes peuvent être tolérées si nous utilisons plus de deux blocs redondants (mais le coût du système augmente également en conséquence).

La section 2.3.5 détaille la manière dont sont réalisés ces différents services.

2.3.5 Analyse de la fusion de données

Notre analyse est divisée en deux parties : un premier cas simple, qui représente une branche de l'architecture proposée (Figure 2.6) et permettant simplement la détection d'une erreur résultant d'une faute matérielle. Ensuite, nous étudions l'architecture complète (Figure 2.5), et les services de rétablissement et de masquage offerts.

Cette étude est en fait analogue à une étude formelle du mécanisme de fusion de données. Nous allons déterminer mathématiquement les paramètres en vertu desquels nos services de tolérance aux fautes seront garantis. Nous avons délibérément choisi un cas simple pour faciliter notre travail, mais notons qu'une telle analyse peut être très coûteuse à réaliser, voire impossible, pour des applications plus complexes.

2.3.5.1 Cas d'une seule branche (un seul jeu de capteurs)

Dans un premier temps, nous étudions un seul jeu de capteurs (caméra, pression). Ce premier cas représente une branche de notre architecture proposée, comme le montre la Figure 2.6.

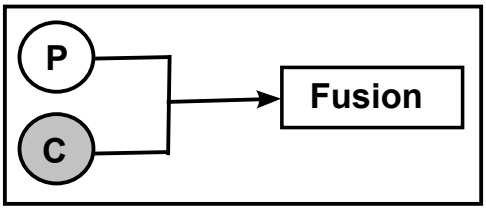

Figure 2.6 – Cas d'une seule branche

2.3.5.1.1 Modélisation Nous avons modélisé ce problème de fusion de données au moyen de la théorie des fonctions de croyance.

- **Cadre de discernement**

  Le cadre de discernement est constitué de deux hypothèses binaires : soit une personne est assise sur la chaise ($\exists$), soit personne n'est assis sur la chaise ($\nexists$). Le cadre de discernement est alors composé uniquement des deux hypothèses ($\exists$) et ($\nexists$), tel que : $\Omega = \{\exists, \nexists\}$.

  A partir de ce cadre de discernement, nous construisons le référentiel de définition constitué de l'ensemble 2^{Ω} des parties de Ω :

  $2^{\Omega} = \{\{\exists\}, \{\nexists\}, \{\exists, \nexists\}, \emptyset\}$

- **Affectation des masses de croyance**

  Considérant $p_i(\exists)$ la sortie du capteur i ($i \in \{P, C\}$), nous avons alors :

$$\begin{cases} p_i(\exists) = 1 : \text{Si quelqu'un est assis sur la chaise.} \\ p_i(\exists) = 0 : \text{Autrement.} \end{cases} \tag{2.1}$$

  Pour chaque type de capteur i est associé un facteur d'affaiblissement $p^i_{trust} \in [0, 1]$. Les masses affectées aux hypothèses $\exists$ et $\nexists$ sont les sorties du capteur $p_i(\exists)$ pondérées par leur facteur d'affaiblissement p^i_{trust} comme indiqué dans (2.2) et (2.3).

$$\begin{cases} m_i(\{\exists\}) = p^i_{trust}.p_i(\exists) \\ m_i(\{\nexists\}) = p^i_{trust}.[1 - p_i(\exists)] \end{cases} \tag{2.2}$$

  D'où

$$\begin{cases} m_i(\{\exists\}) = \begin{cases} p^i_{trust} & si \;\; p_i(\exists) = 1 \\ 0 & si \;\; p_i(\exists) = 0 \end{cases} \\ et \\ m_i(\{\nexists\}) = \begin{cases} p^i_{trust} & si \;\; p_i(\exists) = 0 \\ 0 & si \;\; p_i(\exists) = 1 \end{cases} \end{cases} \tag{2.3}$$

Une fois que les masses $m_i(\{\exists\})$ et $m_i(\{\nexists\})$ sont affectées, la masse restante est associée au cadre de discernement Ω, ce qui modélise l'ignorance :

$$m_i(\{\exists, \nexists\}) = 1 - [m_i(\{\exists\}) + m_i(\{\nexists\})] \tag{2.4}$$

- **Combinaison des masses**

Nous fusionnons les masses des trois hypothèses possibles obtenues à partir des capteurs P et C avec la règle de Dempster comme décrit dans (2.5). Nous obtenons ainsi la croyance globale sur les éléments de 2^{Ω} :

$$\begin{aligned}
m_{P\oplus C}(\emptyset) = &\quad m_P(\{\exists\})m_C(\{\nexists\}) + m_P(\{\nexists\})m_C(\{\exists\}) \\
m_{P\oplus C}(\{\exists\}) = &\quad \frac{m_P(\{\exists\}).m_C(\{\exists\})+m_P(\{\exists\}).m_C(\Omega)+m_P(\Omega).m_C(\{\exists\})}{1-m_{P\oplus C}(\emptyset} \\
m_{P\oplus C}(\{\nexists\}) = &\quad \frac{m_P(\{\nexists\}).m_C(\{\nexists\})+m_P(\{\nexists\}).m_C(\Omega)+m_P(\Omega).m_C(\{\nexists\})}{1-m_{P\oplus C}(\emptyset} \\
m_{P\oplus C}(\Omega) = &\quad 1 - [m_{P\oplus C}(\{\exists\}) + m_{P\oplus C}(\{\nexists\})]
\end{aligned} \tag{2.5}$$

- **Décision**

La décision est basée sur la probabilité pignistique $BetP$, calculée comme suit :

$$\begin{aligned}
BetP(\exists) = &\quad \frac{2m_{P\oplus C}(\{\exists\})+m_{P\oplus C}(\Omega)}{2} \\
BetP(\nexists) = &\quad \frac{2m_{P\oplus C}(\{\nexists\})+m_{P\oplus C}(\Omega)}{2}
\end{aligned} \tag{2.6}$$

Après avoir calculé la probabilité pignistique des deux hypothèses $\exists$ et $\nexists$, la décision est prise en fonction d'une valeur seuil (*threshold*) supérieure strictement à 0.5 comme montré sur la Figure 2.7 selon les deux contraintes suivantes :

- ◇ Si $BetP(\exists) \geq threshold$ alors, nous décidons qu'une personne est assise.
- ◇ Si $BetP(\exists) \leq 1 - threshold$ alors, nous décidons que personne n'est assis.

2.3.5.1.2 Analyse des résultats Dans ce qui suit nous présentons les résultats d'analyse de notre étude, en exposant les différents services de tolérance aux fautes assurés.

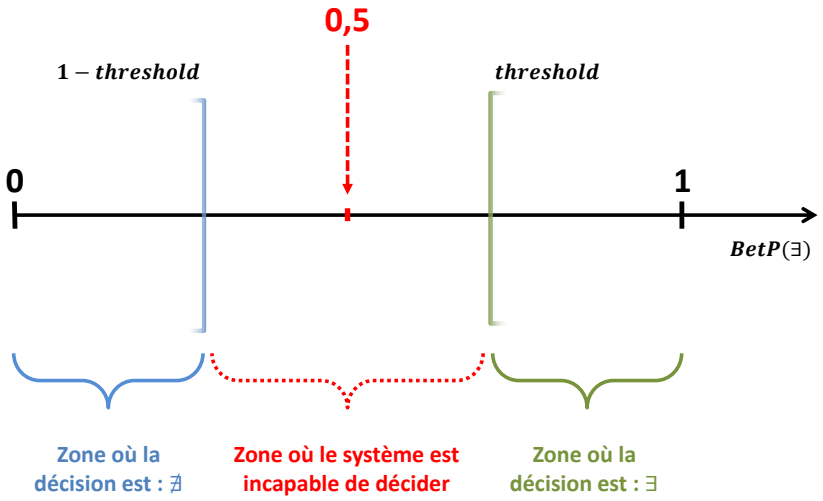

Figure 2.7 – Principe de décision

2.3.5.1.3 Principe de détection La détection de fautes dans ce système de perception est basée sur l'analyse des trois paramètres impliqués au niveau de la fusion de données : les facteurs d'affaiblissement p^P_{trust} et p^C_{trust}, ainsi que le seuil de décision *threshold*. D'autres paramètres pourraient être également étudiés, comme l'opérateur de combinaison ou le critère de décision.

Notre principe de détection des fautes est le suivant : si les deux capteurs ont la même valeur, aucune erreur n'est présente et nous voulons toujours arriver à une décision (la valeur correcte retournée par les capteurs). Si les deux capteurs ont des valeurs différentes, une erreur est présente et nous voulons toujours être incapables de décider. Ainsi, si le système décide, cela signifie qu'aucune erreur n'est présente, et s'il ne décide pas cela signifie qu'une erreur est présente. Pour respecter ce principe, les quatre conditions suivantes doivent êtres assurées.

1. En l'absence de fautes

   - **condition 1 :** Quand il y a effectivement une personne assise sur la chaise, on veut que le système décide toujours qu'il y a bien une personne assise sur la chaise.

$$\begin{cases} P_P(\exists) = 1 \\ P_C(\exists) = 1 \end{cases} \Leftrightarrow BetP(\exists) \geq threshold$$

   - **condition 2 :** Quand il n'y a personne sur la chaise, on veut que le système décide toujours qu'il n'y a personne sur la chaise.

$$\begin{cases} P_P(\exists) = 0 \\ P_C(\exists) = 0 \end{cases} \Leftrightarrow BetP(\exists) \leq 1 - threshold$$

2. En présence de fautes

   - **Condition 3 :** Quand il y a une personne assise sur la chaise et que les deux capteurs donnent des résultats contraires, on veut que le système ne prenne pas de décision.

$$\begin{cases} P_P(\exists) = 1 \\ P_C(\exists) = 0 \\ \text{avec C défaillant} \\ ou \\ P_P(\exists) = 0 \\ P_C(\exists) = 1 \\ \text{avec P défaillant} \end{cases} \Leftrightarrow 1 - threshold < BetP(\exists) < threshold$$

   - **Condition 4 :** Quand il n'y a personne sur la chaise, et que les deux capteurs donnent des résultats contraires, on veut que le système ne prenne pas de décision.

$$\begin{cases} P_P(\exists) = 1 \\ P_C(\exists) = 0 \\ \text{avec P défaillant} \\ ou \\ P_P(\exists) = 0 \\ P_C(\exists) = 1 \\ \text{avec C défaillant} \end{cases} \Leftrightarrow 1 - threshold < BetP(\exists) < threshold$$

2.3.5.1.4 Service de détection Pour pouvoir détecter des fautes dans le système, les quatre conditions citées précédemment doivent êtres satisfaites simultanément. Dans les figures qui suivent, nous allons présenter la probabilité pignistique associée à l'hypothèse $\{\exists\}$ en fonction des facteurs d'affaiblissement p^P_{trust} et p^C_{trust}, afin d'étudier le domaine des facteurs d'affaiblissement vérifiant ces quatre conditions. Nous considérons d'abord une valeur seuil *threshold* de 0,8.

Dans la figure 2.8, l'axe des X représente le facteur p^P_{trust} associé au capteur de pression et l'axe Y le facteur d'affaiblissement p^C_{trust} attribué à la camera. La zone

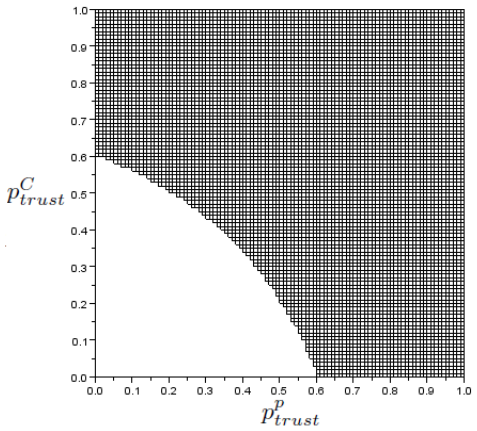

Figure 2.8 – Couples p^i_{trust} satisfaisant la condition 1

grise représente la zone où la probabilité pignistique $BetP(\exists) \geq 0.8$ lorsque les deux capteurs délivrent la même sortie 1, et ainsi respecte la première condition.

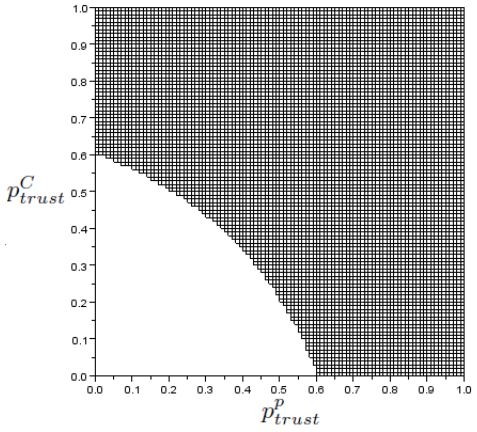

Figure 2.9 – Couples p^i_{trust} satisfaisant la condition 2

La zone grise de la figure 2.9 représente la zone où la probabilité pignistique $BetP(\exists) \leq 0.2$ lorsque les deux capteurs délivrent la même sortie 0, et ainsi respecte la seconde condition. On peut noter que les figures 2.8 et 2.9 sont identiques, en raison de la symétrie de notre problème entre capteurs de pression et caméra.

Les figures 2.10 et 2.11 montrent les facteurs d'affaiblissement (p^P_{trust}, p^C_{trust}) qui répondent aux conditions 3 et 4 respectivement ; la zone grise de ces figures définit la zone où $0.2 < BetP(\exists) < 0.8$. En particulier la figure 2.10 montre la situation où il y a quelqu'un sur la chaise et le système ne décide pas car les capteurs ne fournissent pas les mêmes résultats, tandis que la figure 2.11 montre la deuxième situation où personne n'est assis sur la chaise et le système ne décide pas pour la même raison. Comme avant, par symétrie de notre problème, les surfaces satisfaisant à la troisième et quatrième conditions sont identiques, comme on peut le voir dans

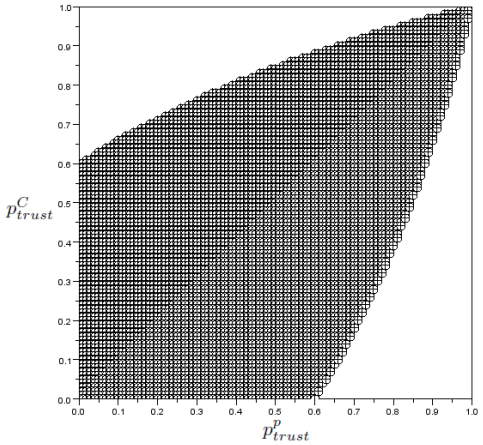

Figure 2.10 – Couples p^i_{trust} satisfaisant la condition 3

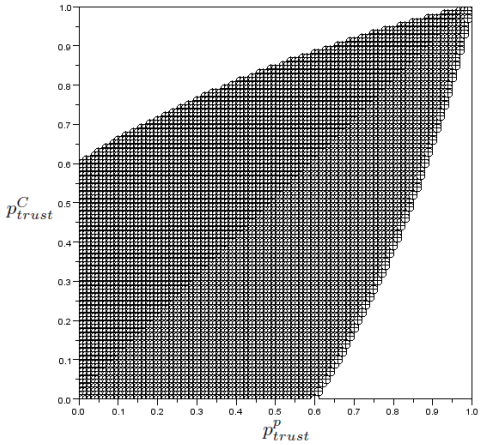

Figure 2.11 – Couples p^i_{trust} satisfaisant la condition 4

les figures (2.10 et 2.11). Dans ces deux figures la zone blanche en haut à gauche représente la zone pour laquelle la sortie du capteur caméra décide le résultat de la fusion : la confiance associée à la caméra est telle que sa sortie impose le résultat de la fusion quelle que soit la sortie du capteur de pression. De même, la zone en bas à droite représente la zone pour laquelle la sortie du capteur pression décide seul le résultat de la fusion.

Finalement, les couples (p^P_{trust}, p^C_{trust}) qui détectent une erreur pour une valeur de seuil de 0,8 sont ceux qui respectent les quatre conditions ci-dessus, et donc le résultat de l'intersection des surfaces des Figures 2.8, 2.9, 2.10 et 2.11. Ces paires sont présentés dans la figure 2.12.

De la même manière que nous avons étudié les couples (p^P_{trust}, p^C_{trust}) pour le service de détection proposé pour $threshold = 0.8$, on peut envisager d'autres valeurs de seuils $thresholds$, et proposer des triplets (p^P_{trust}, p^C_{trust}, $threshold$)

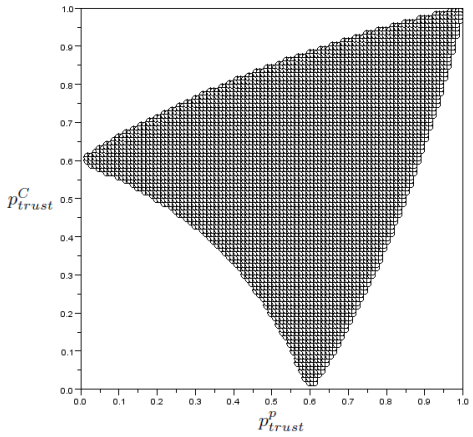

Figure 2.12 – Service de détection pour un seuil de 0.8

assurant le service de détection. Ces triplets sont présentés dans la figure 2.13. Ils se transforment d'une bissectrice ($p^P_{trust} = p^C_{trust}$) pour le seuil threshold = 0.5 aux deux axes ($p^P_{trust} = 1$ et $p^C_{trust} = 1$) pour le seuil $threshold = 1$.

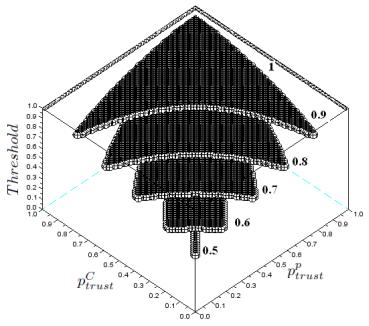

Figure 2.13 – Services de détection pour des seuils variables

Comme mentionné précédemment, seuls les paramètres de modélisation et de décision ont été présentés, mais d'autres paramètres pourraient également être étudiés, comme les paramètres de combinaison en étudiant l'impact de différents opérateurs.

2.3.5.2 Cas de deux branches (Double jeu de capteurs)

Nous avons vu dans la section 2.3.5.1 comment assurer la détection d'erreur avec un seul jeu de capteurs compétitifs. Pour fournir d'autres services tels que le rétablissement du système ou le masquage d'erreur, la duplication des capteurs matériels est nécessaire. Dans la section suivante, nous étudions une architecture de fusion de données complète (Figure 2.5) utilisant deux caméras et deux capteurs de pression, et nous analysons la tolérance aux fautes fournie.

2.3.5.2.1 Modélisation Dans cette architecture, le cadre de discernement est le même que dans le cas simple, et les masses de croyance associées aux hypothèses du problème sont les résultats des deux blocs de fusion F_1 et F_2. La combinaison de ces résultats est effectuée au niveau de *bloc de fusion globale FG* avec la règle de Demspter selon les équations suivantes :

$$\begin{cases} m_{FG}(\emptyset) = & m_{F1}(\{\exists\})m_{F2}(\{\nexists\}) + m_{F1}(\{\nexists\})m_{F2}(\{\exists\}) \\ m_{FG}(\{\exists\}) = & \frac{m_{F1}(\{\exists\}).m_{F2}(\{\exists\})+m_{F1}(\{\exists\}).m_{F2}(\Omega)+m_{F1}(\Omega).m_{F2}(\{\exists\})}{1-m_{GF}(\emptyset)} \\ m_{FG}(\{\nexists\}) = & \frac{m_{F1}(\{\nexists\}).m_{F2}(\{\nexists\})+m_{F1}(\{\nexists\}).m_{F2}(\Omega)+m_{F1}(\Omega).m_{F2}(\{\nexists\})}{1-m_{GF}(\emptyset)} \\ m_{FG}(\Omega) = & 1 - [m_{FG}(\{\exists\}) + m_{FG}(\{\nexists\})] \end{cases} \tag{2.7}$$

2.3.5.2.2 Service de détection Comme dans le cas simple, nous devons satisfaire les quatre conditions précédentes.

Pour une valeur seuil *threshold* de 0.8, la figure 2.14 montre les paires de facteurs d'affaiblissement (p^P_{trust}, p^C_{trust}) qui respectent la première condition, c'est-à-dire les paires pour lesquelles le système décide correctement sur la présence d'une personne lorsque tous les quatre capteurs fournissent la même sortie 1 et aucun capteur n'est erroné.

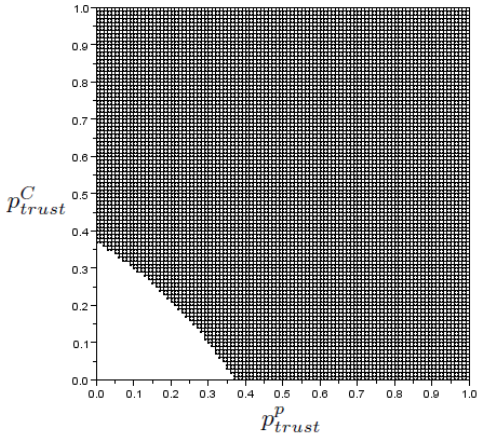

Figure 2.14 – Couples p^i_{trust} satisfaisant la condition 1

La figure 2.15 montre les paires de facteurs d'affaiblissement (p^P_{trust}, p^C_{trust}) qui respectent la deuxième condition : ce sont les paires pour lesquelles le système décide correctement l'absence d'une personne lorsque tous les capteurs fournissent une

même sortie 0 et aucun capteur n'est erroné.

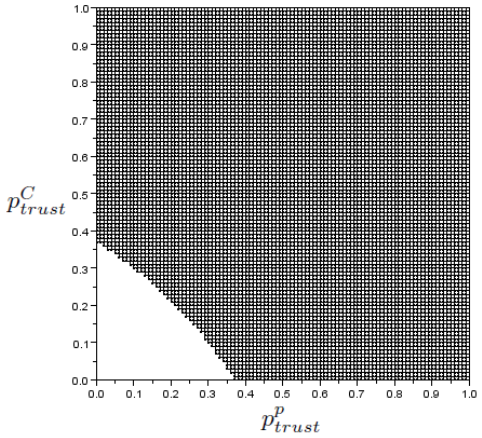

Figure 2.15 – Couples p^i_{trust} satisfaisant la condition 2

Les figures 2.16 et 2.17 montrent les paires de facteurs d'affaiblissement (p^P_{trust}, p^C_{trust}) qui respectent la troisième condition : ce sont les paires de facteurs d'affaiblissement pour lesquels le système n'est pas en mesure de se prononcer sur la présence d'une personne quand exactement un capteur est erroné. Il y a deux figures différentes pour ce cas (contrairement au cas de branche unique montré sur les figures 2.10 et 2.11) parce que nous avons un double jeu de capteurs (deux caméras C_1, C_2 et deux capteurs de pression P_1, P_2), et la situation n'est plus la même si une caméra ou un capteur de pression tombe en panne. En particulier, une camera est défectueuse dans la figure 2.16, et à l'opposé un capteur de pression est défectueux dans la figure 2.17.

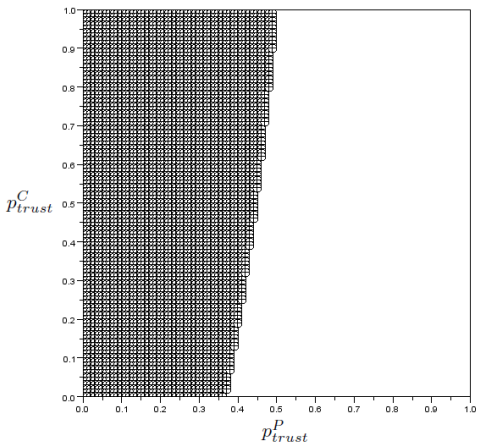

Figure 2.16 – Couples p^i_{trust} satisfaisant la condition 3 avec une caméra défaillante

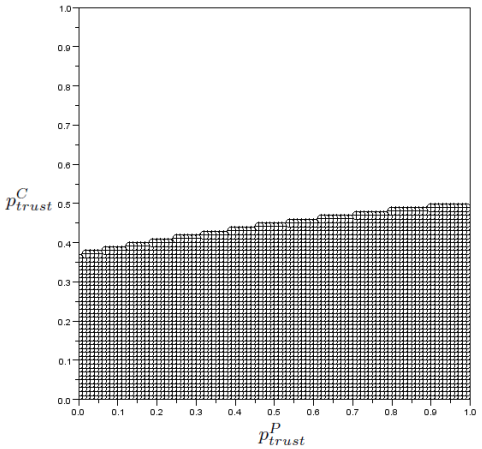

Figure 2.17 – Couples p^i_{trust} satisfaisant la condition 3 avec un capteur pression défaillant

Finalement les couples (p^P_{trust}, p^C_{trust}) montrés sur la figure 2.18 et 2.19 respectant la quatrième condition sont les mêmes que pour la troisième condition (Figures 2.16, 2.17).

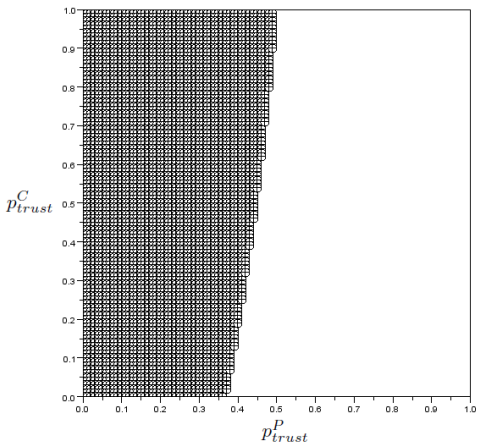

Figure 2.18 – Couples p^i_{trust} satisfaisant la condition 4 avec une caméra défaillante

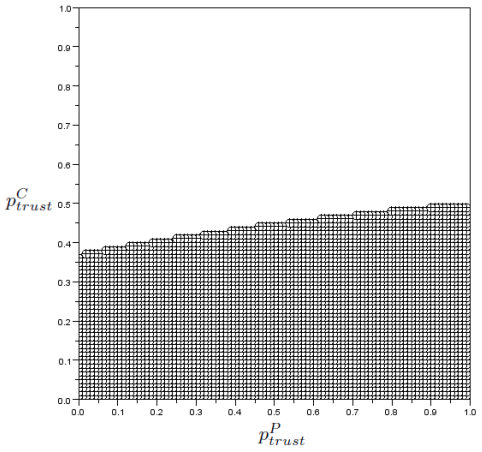

Figure 2.19 – Couples p^i_{trust} satisfaisant la condition 4 avec un capteur pression défaillant

Dans les figures 2.16, 2.17, 2.18 and 2.19 nous pouvons noter que la zone blanche représente les couples (p^P_{trust}, p^C_{trust}) pour lesquels le système prend une décision correcte en dépit d'une erreur dans un capteur. Cependant, nous cherchons ici à détecter l'erreur et de ne pas décider correctement, et ce domaine est donc exclu des couples satisfaisant la détection.

Comme dans le cas simple, pour détecter une erreur dans le système considéré les quatre conditions doivent être remplies. L'intersection des figures précédentes montrée sur la figure 2.20 donne les couples (p^P_{trust}, p^C_{trust}) qui permettent le service de détection.

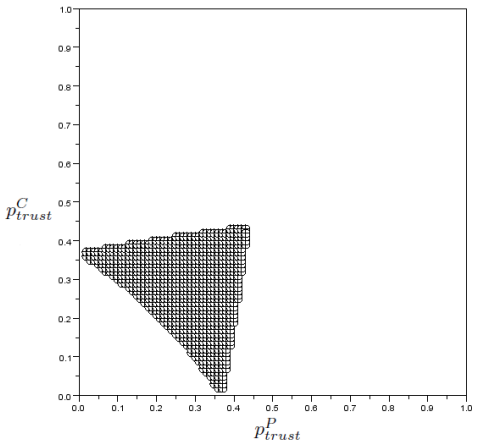

Figure 2.20 – Service de détection pour un seuil 0.8

La figure 2.21 présente les triplets p^P_{trust}, p^C_{trust}, $threshold$) fournissant le service de détection pour l'architecture complète (Figure 2.5). Il se transforme à partir de l'unique point ($p^P_{trust} = p^C_{trust} = 0$) pour un seuil de 0.5 au deux axes ($p^P_{trust} = 1$ et $p^C_{trust} = 1$) pour un seuil de 1, ce cas limite semble de peu d'intérêt pratique.

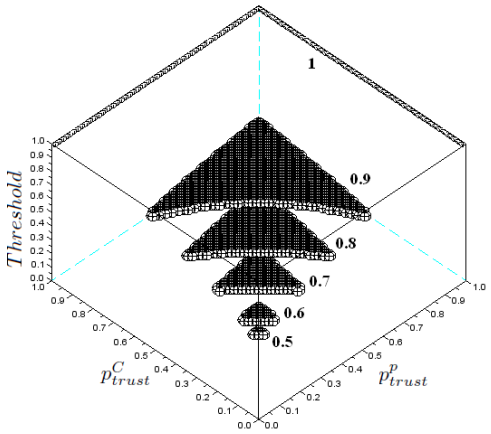

Figure 2.21 – Services de détection pour des seuils variables

2.3.5.2.3 Service de recouvrement par compensation Ainsi que précisé dans le chapitre 1 la compensation d'erreurs consiste à utiliser les redondances présentes dans le système pour déterminer le résultat correct en dépit de la présence d'une erreur.

Pour tolérer une erreur matérielle, l'architecture complète de la figure 2.5 doit tout d'abord détecter la présence d'une erreur, puis déterminer la sortie correcte à renvoyer. La détection est assurée de la même manière que précédemment : on choisit les paramètres satisfaisant le service de détection (zone grise) donnés en figure 2.20. Une fois l'erreur détectée $Bet(\exists) \in [1 - threshold, threshold]$, et sous l'hypothèse d'une erreur unique, il nous faut encore trouver laquelle des deux branches (F_1 ou F_2) contient le capteur erroné. Si les deux branches respectent les critères de détection d'erreurs de la section 2.3.5.1, cela est facile : la branche contenant le capteur erroné n'aura pas réussi à décider, et l'autre branche aura abouti à la sortie correcte.

Le couple de facteurs d'affaiblissement (p^P_{trust}, p^C_{trust}) permettant le service de rétablissement par compensation doit donc :

- permettre de détecter une erreur dans le cas complexe (double jeu de capteurs),
- et permettre de détecter une erreur dans le cas simple (un seul jeu de capteurs).

Il est ainsi l'intersection des figures 2.12 et 2.20, présentée en figure 2.22.

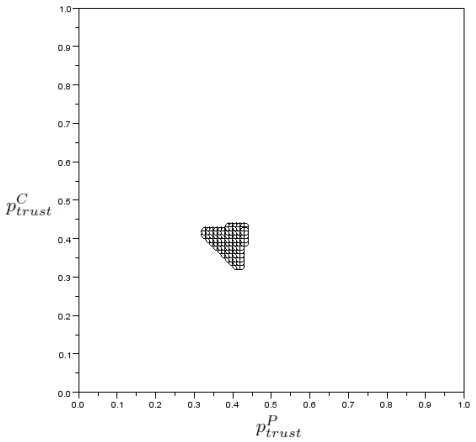

Figure 2.22 – Service de compensation pour un seuil 0.8

Les triplets (p^P_{trust}, p^C_{trust}, $threshold$) qui fournissent la compensation d'erreur sont donnés dans la figure 2.23. il se transforme à partir de l'unique point ($p^P_{trust} = p^C_{trust} = 0$) pour un seuil de 0,5 au point ($p^P_{trust} = p^C_{trust} = 1$) pour un seuil de 1.

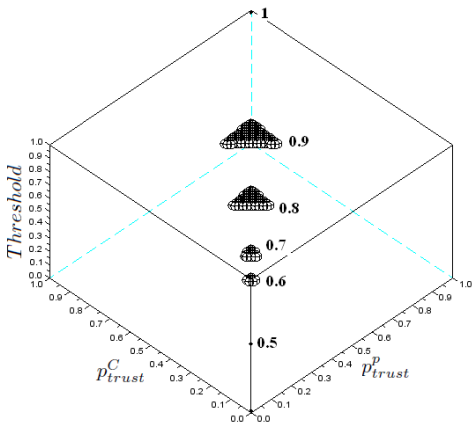

Figure 2.23 – Services de compensation pour des seuils variables

L'algorithme 2.2 montre comment le service de recouvrement par compensation est assuré dans l'architecture complète de la figure 2.5.

Cet algorithme est exécuté de la manière suivante : il vérifie si la fusion globale rend une décision sur une des hypothèses ($\exists$, $\nexists$). Si tel est le cas, alors le système ne contient pas d'erreur. Sinon, il cherche laquelle des branches de fusion (F_1 ou F_2) ne décide pas afin d'en déduire la branche erronée. Il prend alors la sortie de l'autre branche comme sortie correcte du système.

Algorithm 2.2: Algorithme de compensation d'erreur

Input: Résultats de la fusion : probabilités pignistiques résultant de blocs de fusion F_1, F_2 et FG
Output: E : Branche erronée;
O : Sortie du système;

```
1  Function Decision(Résultats de la fusion) : Booléen
2   BeginFunction
3  if ((BetP({∃}) ≥ Threshold) Or (BetP({∃}) ≤ 1 − Threshold)) then
4  |  return(VRAI);
5  else
6  |  return(FAUX);
7  end
8  EndFunction
9  if (Decision(BetP(∃) de FG) = VRAI) then
10 |  E ← {∅}; O ← {FG};
11 else
12 |  Erreur detectée;
13 |  if ((Decision(BetP(∃) de F1) = VRAI) et (Decision(BetP(∃) de F2) =
   |  FAUX)) then
14 |  |  E ← {F2}; O ← {F1};
15 |  else
16 |  |  E ← {F1}; O ← {F2};
17 |  end
18 end
```

2.3.5.2.4 Service de masquage L'architecture de la figure 2.5 peut tolérer une erreur matérielle (sortie d'un capteur erroné) sans la détecter. Pour ce faire il faut :

- que le système ait un comportement correct en l'absence d'erreur, ce qui revient à satisfaire les deux premières conditions indiquées dans les figures 2.14 et 2.15 : Le système décide de la présence d'une personne sur la chaise quand tous les capteurs délivrent une sortie correcte égale à 1 (qui correspond à la première condition), et il décide que personne n'est assis sur la chaise quand tous les capteurs délivrent une sortie correcte égale à 0 (ce qui correspond à la deuxième condition).

- que le système aboutisse au résultat correct malgré la présence d'une erreur. Ceci correspond aux couples (p^P_{trust}, p^C_{trust}) des figures 2.24 et 2.25 facilement retrouvées à partir des figures 2.16 et 2.17 ou 2.18 et 2.19. pour lesquels on décide de la présence (respectivement l'absence) d'une personne sur la chaise à bon escient.

En effet, la figure 2.24 correspond au cas où les deux capteurs de pression fournissent la même valeur correcte alors que les caméras délivrent deux valeurs

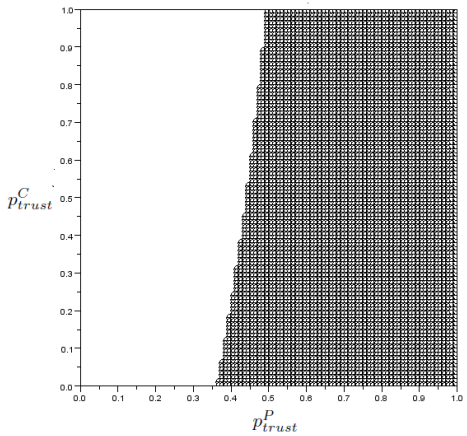

Figure 2.24 – Résultat correct du système malgré l'erreur d'une caméra

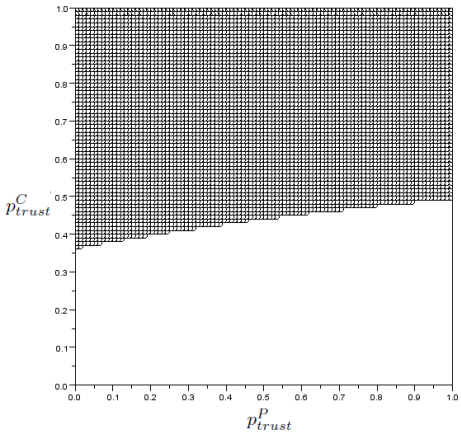

Figure 2.25 – Résultat correct du système malgré l'erreur d'un capteur de pression

contradictoires comme du faite que l'une d'entre elles est défectueuse. Avec ces facteurs d'affaiblissement, la valeur des capteurs de pression, par conséquent masquent la caméra défectueuse. La Figure 2.25 correspond au cas inverse, où le système masque une faute sur l'un des capteurs de pression.

Les couples (p^P_{trust}, p^C_{trust}) permettant le masquage de fautes sont donc ceux appartenant à l'intersection des figures 2.14, 2.24 et 2.25, représentée dans la figure 2.26.

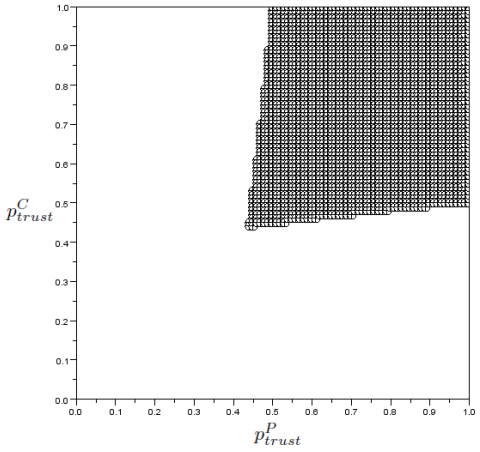

Figure 2.26 – Service de masquage pour un seuil 0.8

La figure 2.27 présente les triplets (p^P_{trust}, p^C_{trust}, $threshold$) satisfaisant le service de masquage, il se transforme de l'ensemble du domaine pour un seuil $threshold = 0.5$, à l'unique point $p^P_{trust} = p^C_{trust} = 1$ pour un $threshold = 1$.

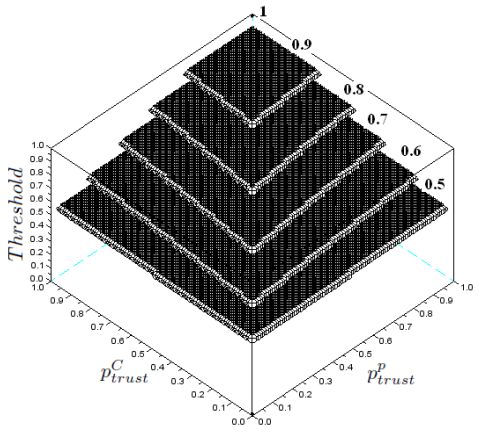

Figure 2.27 – Services de masquage pour des seuils variables

L'algorithme 2.3 montre la trivialité de l'algorithme de masquage : le système n'a pas besoin d'exécuter des actions spécifiques. Comme indiqué précédemment, il est toutefois plus risqué que le mécanisme de compensation, en particulier si un entretien régulier n'est pas effectué.

Algorithm 2.3: Algorithme de masquage

Output: O : Sortie du système
1 $O \leftarrow$ Fusion Globale();
2 Return(O);

2.3.5.3 Bilan

La figure 2.28 résume les différents services de tolérance aux fautes offerts par l'architecture complète de la figure 2.5. Ces services sont :

- *La détection d'erreur :* détecte une erreur matérielle.
- *Le rétablissement :* détecte et tolère une erreur matérielle.
- *Le masquage :* tolère, sans détection, une erreur matérielle.

Ces services peuvent être offerts dès lors que la fusion de données respecte les conditions de la figure 2.28, identifiées lors d'une analyse formelle de son modèle. Dans le cas volontairement simple de notre étude théorique, cette analyse n'a pas montré de grandes difficultés. Cela pourrait ne pas être le cas dans des applications plus complexes où cette analyse pourrait même ne pas avoir de solutions. Seule une vraie diversification logicielle permettrait alors d'apporter des solutions en termes de tolérance aux fautes, et ainsi fournir des solutions fiables, et l'approche de *duplication / comparaison* serait donc nécessaire.

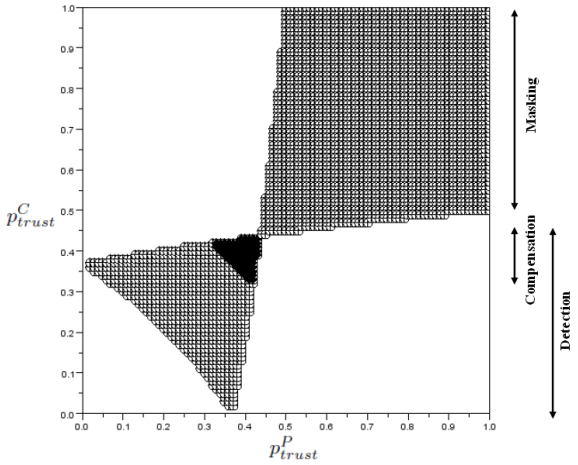

Figure 2.28 – Services de tolérance au fautes en fusion de données pour un seuil de 0.8

2.4 Comparaison entre les deux approches proposées : la fusion de données et la duplication-comparaison

Avant de conclure ce chapitre, nous présentons dans cette section une étude qualitative visant à comparer la tolérance aux fautes fournie par la fusion de données (Figure 2.2) et celle fournie par la duplication / comparaison (Figure 2.1) :

Le principal avantage de la fusion de données (seule approche utilisée dans la littérature avec critères définis empiriquement sans validation formelle) est de détecter et tolérer une erreur matérielle sans la nécessité de comparer directement les sorties des capteurs. En outre, même si ce n'était pas l'objet de ce travail, nous pouvons soutenir que le mécanisme de fusion globale de la figure 2.2 contribue à la fonction du système en fournissant des données plus précises qu'une simple branche F_1 ou F_2. Cependant, il nécessite une étude analytique préalable qui peut être complexe et peut ne pas avoir des solutions dans les applications complexes. Sans cette analyse, le comportement de l'algorithme de fusion ne peut pas être garanti, ce qui pour nous, rend inacceptable une telle approche pour des systèmes critiques. De plus bien qu'il ne soit pas possible de détecter les fautes logicielles, une telle analyse formelle nous donne une certaine confiance dans le comportement correct du système (bien que les erreurs de programmation puissent encore être présentes).

La duplication / comparaison permet de détecter et de tolérer non seulement les erreurs matérielles liées aux capteurs, mais également des erreurs logicielles dans les algorithmes de fusion, cependant elle exige toujours des mécanismes pour comparer les valeurs des capteurs. Pour rétablir le système d'une erreur logicielle, trois mécanismes de fusion diversifiés sont également nécessaires.

Notons que ces services ne sont disponibles pour l'approche de fusion de données que dans les stratégies compétitives et redondantes ou complémentaires présentées dans le chapitre précédent. En effet, la tolérance aux fautes ne peut pas être assurée sans une certaine forme de redondance. Toutefois, les services de tolérance aux fautes sont disponibles dans toutes les stratégies pour l'approche de duplication / comparaison comme nous avons deux branches redondantes indépendantes à considérer.

Le tableau 2.2 résume les services de tolérances aux fautes offerts par les deux architectures des figures 2.2 et 2.1 (fusion de données et duplication-comparaison respectivement).

Mécanisme / **Service**	**Fusion de données**	**Duplication-Comparaison**
Détection de faute matérielle	**Oui**	**Oui** : nécessite des mécanismes de comparaison des sorties de capteurs
Détection de faute logicielle	**Non** : Pas nécessaire étant donné que la fusion de données a été validée par l'analyse formelle	**Oui** : nécessite des mécanismes de comparaison des sorties de capteurs
Tolérance de faute matérielle	**Oui**	**Oui** : exige des mécanismes de comparaison des sorties du capteurs et que le conflit dans la fusion de données soit représentatif
Tolérance de faute logicielle	**Non** : Pas nécessaire étant donné que la fusion de données a été validée par l'analyse formelle	**Oui** : exige un troisième bloc de fusion de données
Masquage	**Oui**	**Non** : mais éventuellement avec un troisième bloc de fusion de données

Tableau 2.2 – Services de tolérance aux fautes par la fusion de données et la duplication de comparaison

2.5 Conclusion

Dans ce chapitre nous avons présenté deux approches de tolérance aux fautes en fusion de données : la première est basée sur la méthode de duplication-comparaison, et la seconde est intégralement basée sur la fusion de données sans aucun mécanisme de tolérance aux fautes. Elle est utilisée dans la littérature mais, pour nous requiert une analyse formelle de la fusion de données pour garantir son comportement par

exemple comme celle décrite dans ce chapitre. Les deux approches assurent des services de détection, de rétablissement et du masquage de fautes dans la perception multicapteurs :

1. **La duplication / comparaison** : assure la tolérance aux fautes de la manière suivante :
   - *La détection de faute* est assuré par la comparaison des sorties des deux blocs de fusion de données implémentés.
   - *Le rétablissement du système* est garanti par la comparaison des sortie des capteurs et l'analyse de résidu (conflit) entre les deux blocs de fusion. Le rétablissement d'une erreur logicielle peut être assuré par la diversification des algorithmes de fusion et la méthode de vote majoritaire.
2. **Étude formelle de la fusion de données** les services de tolérances aux fautes fournis par cette approche sont assurés de la manière suivante :
   - *La détection de faute* est réalisée par l'analyse de modèle de fusion qui cherche à trouver les paramètres internes à la fusion tels que les facteurs d'affaiblissement des capteurs et le critère de décision garantissant le service de détection.
   - *Le rétablissement du système* consiste a choisir les paramètres (facteurs d'affaiblissement) satisfaisant les conditions de détection de fautes dans les deux configurations (un seul jeu de capteur, double jeu de capteur).

Nous avons réalisé une étude comparative des deux approches proposées, et nous avons constaté que l'approche basée sur l'analyse formelle de la fusion de données bien qu'elle donne une certaine confiance dans le comportement correct du système de perception, peut être très difficile à réaliser voire impossible dans les applications complexes. L'approche de duplication/ comparaison exige la comparaison des sorties des capteurs et la définition des seuils pour la détection et le rétablissement, mais réalisable et assure en plus la tolérance aux fautes logicielles.

Le prochain chapitre implémente l'architecture générique de duplication comparaison dans le cadre d'application de localisation des robots mobiles.

Application : filtres de Kalman pour la localisation des robots mobiles

Sommaire

3.1 Introduction

Dans ce chapitre nous présentons un exemple d'application de l'architecture générique de tolérance aux fautes basée sur la duplication-comparaison exposée dans la section 2.2 de chapitre 2. Nous appliquons cette architecture pour la localisation d'un véhicule utilisant la fusion de données par filtrage de Kalman.

Ce chapitre est organisé comme suit : Nous décrivons d'abord notre véhicule de laboratoire ainsi que ses capteurs. Puis nous présentons les deux modèles cinématiques des robots mobiles (char et tricycle) avant de détailler les deux filtres implémentés. Enfin nous exposons les services de tolérance aux fautes offerts par notre architecture sur ce cas d'étude de localisation, en détaillant les algorithmes de détection de fautes et de rétablissement du système.

3.2 Plateforme Matérielle : Véhicule Carmen

Le véhicule utilisé pour l'expérimentation de ces travaux est le véhicule Carmen (Citroën C5 Break montré dans la Figure 3.1) du laboratoire Heudiasyc. Ce véhicule

est instrumenté pour la perception et est équipé de plusieurs capteurs proprioceptifs (capteurs utilisés pour percevoir l'état interne du robot) et extéroceptifs (capteurs utilisés pour percevoir l'environnement du robot) : un lidar IBEO Alasca XT dans le pare-chocs avant, des capteurs odométriques qui communiquent par une passerelle avec le BUS CAN, un système de positionnement IMU-GPS Novatel SPAN CPT et une caméra Sony. Le véhicule est aussi équipé d'un système de stéréovision Videre, d'un radar 77 GHz, d'un récepteur GPS haute qualité Septentrio PolarX et d'un télémètre laser à balayage à 64 faisceaux Velodyne 64 HEL. Tous ces capteurs sont connectés à un PC via différentes interfaces (Série, Ethernet, USB et Firewire). Dans ce qui suit nous nous basons sur la description du sous ensemble de capteurs utilisés dans nos travaux.

Figure 3.1 – Véhicule expérimental (CARMEN)

3.2.1 Les capteurs de perception utilisés

La perception en robotique mobile a pour but de permettre au robot de recueillir, traiter et mettre en forme des informations qui lui sont utiles pour agir et réagir dans l'environnement qui l'entoure. Elle est ainsi la faculté de détecter et/ou appréhender l'environnement proche ou éloigné du robot. Pour extraire les informations utiles à l'accomplissement de la tâche du robot, il est nécessaire que celui-ci embarque de nombreux capteurs mesurant son état interne ainsi que l'état de l'environnement dans lequel il évolue. Le choix des capteurs dépend bien évidemment de l'application envisagée. Pour notre cas, la localisation du robot dans son environnement, nous utilisons plusieurs capteurs proprioceptifs et extéroceptifs dont dispose le véhicule d'expérimentation. Nous décrivons ces capteurs ainsi que les données fournies dans la suite de cette section.

- **Le GPS** est un système de positionnement conçu par le Département de la Défense des Etats-Unis (DoD) au début des années 1970 à des fins

d'applications de navigation militaires. Ce système est actuellement à la disposition du grand public et largement utilisé pour des applications civiles telles que la localisation et le positionnement. Le système GPS est une solution efficace à presque toutes les applications de localisation dans le domaine de la robotique mobile. Les principales informations fournies par le GPS sont les coordonnées géographiques tridimensionnelles du récepteur (l'utilisateur) à savoir sa longitude, sa latitude, et son altitude de manière continue et instantanée, en tout endroit sur Terre. [Abuhadrous, 2005] résume les avantages et les inconvénients du système de localisation absolue GPS comme suit :

◇ **Avantages**

✓ *Précision à long terme* : En raison d'absence de dérive en GPS, sa précision ne se dégrade pas au fil du temps.

✓ *Position absolue* : Le GPS fournit une position absolue et non pas relative car elle ne dépend pas de conditions initiales.

✓ *Indépendance de l'environnement* : Le système marche jour et nuit, il est fonctionnel sans besoin de conditions spéciales (température, orientation ...)

◇ **Inconvénients**

✓ *Manque de contrôle* : Étant donné que le GPS est un système américain, les utilisateurs européens n'ont aucun contrôle ni aucune garantie légale de bon fonctionnement.

✓ *Problème de multitrajet* : Ce phénomène se produit quand le signal GPS arrive au récepteur après plus d'un trajet à cause de réflexions près du récepteur, par des objets tels que des grands bâtiments ou de grandes surfaces de roche avant qu'il atteigne le récepteur. Ceci augmente le temps de parcours du signal, causant par conséquent une surévaluation du temps de vol et générant par conséquent des erreurs de position.

✓ *Disponibilité faible* : La précision dépend du nombre de satellites visibles par le récepteur (la précision est bien meilleure quand le récepteur voit plus de satellites). Dans les applications dans le domaine automobile, les conditions de visibilité sont souvent dégradées par les bâtiments, les tunnels, l'interférence électronique, ou parfois même un feuillage dense ; ceci peut bloquer la réception du signal, entraînant des erreurs de position voire causant un arrêt temporaire

du système. Typiquement, les unités de GPS ne fonctionneront pas dans des environnements d'intérieur, sous l'eau ou sous terre. Les erreurs dans une ville avec des gorges de bâtiments vont être de l'ordre d'une dizaine de mètres.

- **La centrale inertielle** est un équipement de navigation largement utilisé en robotique mobile. Elle est composée de gyromètres et d'accéléromètres :

  1. **Le gyromètre** est un capteur proprioceptif qui permet de mesurer l'orientation du corps sur lequel il est placé, ceci par rapport à un référentiel fixe et selon un ou deux axes. Monté sur un robot mobile plan, un gyromètre à un axe mesure la vitesse angulaire de ce robot, une intégration de cette mesure permettant d'obtenir l'angle de lacet du robot.
  2. **L'accéléromètre** est un appareil qui mesure l'accélération linéaire, une simple intégration de cette mesure permettant d'avoir la vitesse, et une double intégration donnant la position. Pour avoir une position en trois dimensions, trois accéléromètres mesurant les accélérations linéaires selon chaque axe sont nécessaires.

     Les avantages et les inconvénients de ces capteurs inertiels sont résumés comme suit :

     - ◇ **Avantages** :
       - ✓ *Solution complète* : les centrales inertielles fournissent en sortie position, vitesse, et accélération en translation et rotation.
       - ✓ *Bonne précision à court terme* : si le système est bien initialisé, la précision est très bonne sur une courte durée, avant que les effets de dérive ne commencent à devenir trop forts.
       - ✓ *Haute disponibilité* : Le système ne dépend pas de dispositifs extérieurs, ce qui le rend entièrement autonome.
       - ✓ *Petite taille* : La taille raisonnable des centrales inertielles facilite et motive leur utilisation dans tout type d'application. Leur taille de plus en plus petite, ne pose pas de contrainte particulière de placement pour être embarquée dans tout type de robot.
     - ◇ **Inconvénients** :
       - ✓ *Forte dérive* : les effets de biais engendrent une très forte dérive après la double intégration, ce qui fait que l'imprécision augmente fortement avec le temps.

✓ *Pas d'information absolue* : étant un système de navigation à l'estime, la position et la vitesse sont toujours calculées à partir des conditions initiales.

- **Le capteur odométrique** est un dispositif associé aux roues des robots mobiles. Il permet de mesurer les déplacements des roues pour reconstituer le mouvement global du robot. En partant d'une position initiale connue et en intégrant les déplacements mesurés, on peut ainsi calculer à chaque instant la position courante du véhicule. Les systèmes odométriques possèdent les propriétés suivantes :

  ◇ **Avantages**

  ✓ *Le prix* : peu coûteux, ce qui fait que l'odométrie est actuellement intégrée d'office dans les nouveaux modèles de voitures à travers les capteurs ABS. Leur utilisation pour la navigation n'a pas de charge ou de coût supplémentaire. Cependant, une meilleure précision est souhaitable.

  ✓ *Bonne précision à court terme* : la précision du système est bonne sur une courte période de temps avant que les dérives prennent effet.

  ✓ *Simple à implémenter* : les équations d'odométrie sont simples, donc elle est facilement implémentable et peu coûteuse en terme de développement.

  ◇ **Inconvénients**

  ✓ *L'imprécision* : à cause des dérives, patinages des roues, glissements, et autres, les mesures d'angle et de distance accumulent les erreurs avec le temps.

  ✓ *Information en 2D* : l'odométrie mesure la distance parcourue sur le sol. Si la route est inclinée, la distance mesurée est oblique.

  ✓ *Pas d'information absolue* : l'odométrie est un système de navigation à l'estime donc on a besoin d'un capteur externe (par exemple le GPS) pour avoir un positionnement absolu.

  [Borenstein and Feng, 1996] classe les sources de toutes les erreurs odométriques en deux catégories : *erreurs systématiques* et *erreurs non-systématiques*.

  ◇ *Erreurs systématiques* : ces erreurs s'ajoutent à chaque itération de localisation. Elles sont dues aux erreurs sur les paramètres mécaniques du

véhicule, ou aux erreurs de mesures. On cite ci-dessous quelques exemples de ce type d'erreurs.

- ✓ Diamètres des roues inégales,
- ✓ Moyenne des deux diamètres de roues différente du diamètre nominal,
- ✓ Désalignement des roues,
- ✓ L'empattement (la distance inter-essieux du véhicule) réel diffère de l'empattement nominal.
- ✓ Fréquence d'échantillonnage inexacte.

◇ *Erreurs non systématiques* : ces erreurs sont liées au contact véhicule / terrain. Elles sont aléatoires et généralement dépendantes des caractéristiques du sol sur lequel roule le robot. Parmi ces erreurs on trouve :

- ✓ roulement sur des sols irréguliers,
- ✓ passage sur des obstacle inattendus,
- ✓ glissement des roues.

3.2.2 Mise en œuvre de l'architecture de tolérance aux fautes

Les détails de l'architecture implémentée sont montrés dans la Figure 3.2. Dans cette application nous implémentons deux branches parallèles, chacune contenant deux blocs de capteurs : le premier pour la perception de la commande et le second pour l'observation de la mesure. Un bloc de fusion de données combine les sorties de ces blocs de capteurs par un filtre de Kalman.

La redondance physique des capteurs de perception, et la diversification logicielle des algorithmes de fusion de données (filtres de Kalman) permet d'assurer les services de tolérance aux fautes suivants :

- *La détection et le recouvrement d'une faute matérielle* : détecte et recouvre une erreur matérielle dans les capteurs de perception (R-WSS (Rear Wheel Speed Sensor), F-WSS (Front Wheel Speed Sensor), INS (Inertial Navigation System), GPS (Global Positiong System)).

- *La détection d'une faute logicielle* : détecte une erreur logicielle dans les algorithmes de filtre de Kalman (FK_1, FK_2).

- *Le recouvrement d'une faute logicielle* : la faute logicielle détectée ne peut être tolérée que par un troisième filtre de Kalman diversifié et la méthode de vote majoritaire, qui ne sont pas mis en place dans cette application.

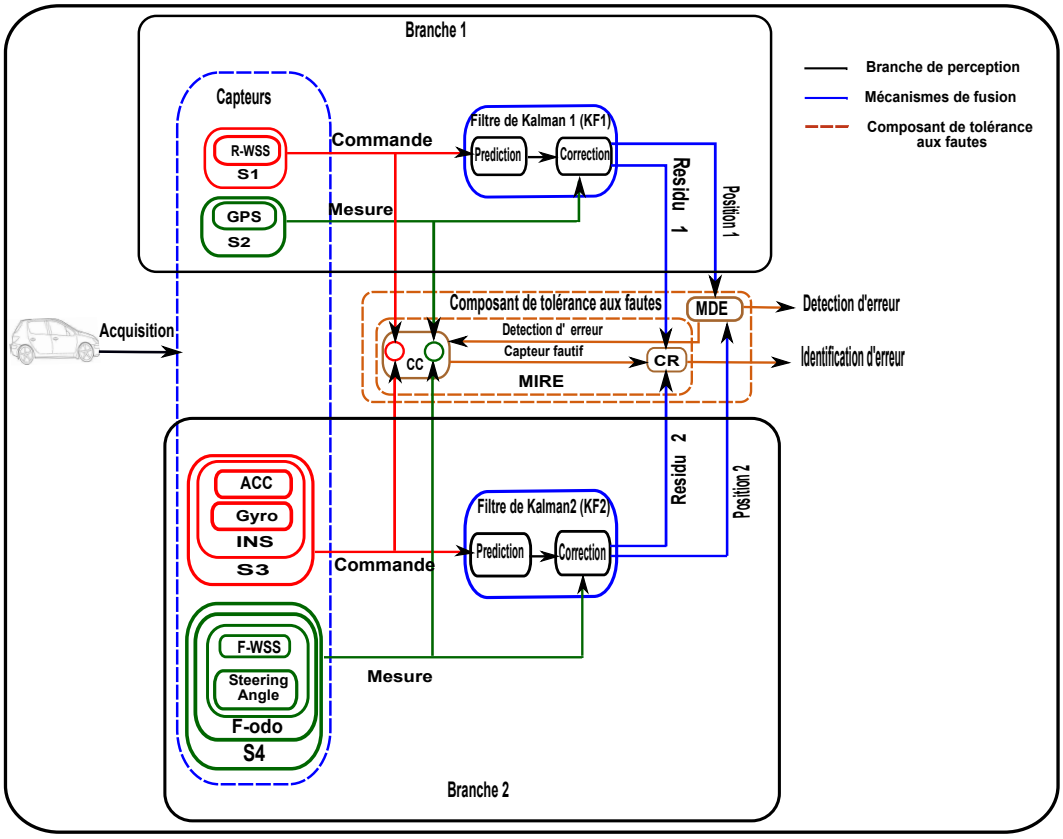

Figure 3.2 – Mise en œuvre de l'architecture pour la localisation du robot en utilisant les filtres de Kalman

3.3 Modèles cinématiques de véhicule

Pour bien comprendre les deux filtres de Kalman implémentés dans ce chapitre, une introduction des modèles cinématiques de véhicule employés dans ces filtres est nécessaire. Cette section présente ces modèles, en rappelant brièvement leurs équations.

3.3.1 Équations de modèle de type tricycle

Dans ce modèle montré en Figure 3.3, le robot est constitué de deux roues arrières fixes sur le même axe et une roue avant centrée orientable placée sur l'axe longitudinal du robot. Le mouvement est conféré au robot par deux actions : la vitesse longitudinale et la direction de la roue orientable.

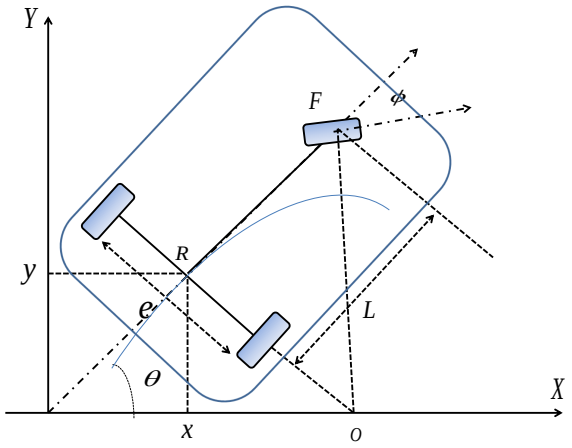

Figure 3.3 – Modèle cinématique de type tricycle

L'état du véhicule peut être représenté par (x, y, θ, ϕ) comme l'illustre la figure 3.3, avec :

- (x, y) : la position,
- θ : l'angle de lacet, c'est à dire l'angle que fait le corps du véhicule avec l'axe x horizontal,
- ϕ : l'angle de direction par rapport au corps du véhicule c'est-à-dire l'angle de braquage de la roue.

Deux grandeurs caractérisent par ailleurs le véhicule

- L : la distance entre les essieux du véhicule (la distance entre R et F dans la figure).
- e : la distance entre les deux roues arrière.

Selon que les capteurs odométriques sont montés sur les roues arrière ou sur les roues avant du véhicule, deux modèles cinématiques existent. Étant donné que notre véhicule dispose des encodeurs arrière et avant, nous exploitons dans notre architecture cette réplication afin d'augmenter le niveau de redondance physique pour des fins de tolérance aux fautes. Ci-dessous sont présentés ces deux modèles tricycles :

3.3.1.1 Modèle tricycle utilisant les roues arrière

Si les capteurs odométriques sont montés sur les roues arrière de la voiture, le modèle cinématique est décrit par :

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} cos\theta \\ sin\theta \\ \frac{tan\phi}{L} \end{bmatrix} .V_R \tag{3.1}$$

La forme discrète de ces équations est :

$$\begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix} = \begin{bmatrix} x_{k-1} \\ y_{k-1} \\ \theta_{k-1} \end{bmatrix} + \begin{bmatrix} \Delta t.cos\theta_{k-1} \\ \Delta t.sin\theta_{k-1} \\ \Delta t.\frac{tan\phi_{k-1}}{L} \end{bmatrix} .V_{R_{k-1}} \tag{3.2}$$

Où Δt est la période d'échantillonnage, et V_R la vitesse longitudinale des roues arrière du véhicule, qui peut être calculée comme suit :

$$V_R = \frac{V_{RR} + V_{RL}}{2} \tag{3.3}$$

Avec V_{RR} et V_{RL} respectivement la vitesse de la roue arrière droite et la vitesse de la roue arrière gauche fournies par les capteurs odométriques arrière (R-WSS) associés à chacune des roues arrière.

3.3.1.2 Modèle tricycle utilisant la roue avant

Si les capteurs odométriques sont montés sur la roue avant de la voiture, le modèle cinématique est décrit par :

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} cos\theta.cos\phi \\ sin\theta.sin\phi \\ \frac{sin\phi}{L} \end{bmatrix} .V_F \tag{3.4}$$

La forme discrète de ces équations est :

$$\begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix} = \begin{bmatrix} x_{k-1} \\ y_{k-1} \\ \theta_{k-1} \end{bmatrix} + \begin{bmatrix} \Delta t.cos\theta_{k-1}.cos\phi_{k-1} \\ \Delta t.sin\theta_{k-1}.sin\phi_{k-1} \\ \Delta t.\frac{sin\phi_{k-1}}{L} \end{bmatrix} .V_{F_{k-1}} \tag{3.5}$$

Où V_F est la vitesse de la roue avant du véhicule fournie par les capteurs odométriques avant (F-WSS).

3.3.2 Équations de modèle de type char

Dans ce type de modèle montré dans la Figure 3.4, le robot dispose des roues gauche et droite, mais pas de roue orientable comme on le trouve dans le modèle

présenté précédemment. Les encodeurs associés à ces roues permettent de mesurer les déplacements élémentaires de ces dernières. Le calcul de pose (position et orientation) du robot se fait à partir de ces mesures en utilisant les équations cinématiques décrites dans 3.7.

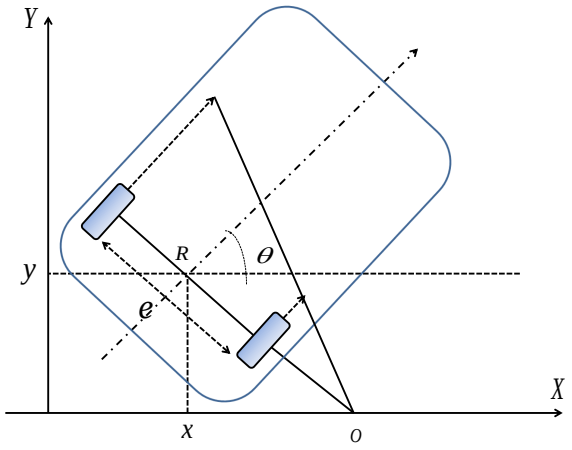

Figure 3.4 – Modèle cinématique de type char

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} cos(\theta) & 0 \\ sin(\theta) & 0 \\ 0 & 1 \end{bmatrix} . \begin{bmatrix} V_R \\ V_R \\ \omega_R \end{bmatrix} \tag{3.6}$$

La forme discrète de ce modèle est la suivante :

$$\begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix} = \begin{bmatrix} x_{k-1} \\ y_{k-1} \\ \theta_{k-1} \end{bmatrix} + \begin{bmatrix} \Delta t.cos(\theta_k) & 0 \\ \Delta t.sin(\theta_k) & 0 \\ 0 & \Delta t \end{bmatrix} . \begin{bmatrix} V_{R_{k-1}} \\ \omega_{R_{k-1}} \end{bmatrix} \tag{3.7}$$

La vitesse angulaire $\dot{\theta} = \omega_R$ peut être calculée directement à partir des sorties des encodeurs arrière V_{RR} et V_{RL} suivant l'équation (3.8) :

$$\dot{\theta} = \frac{V_{RR} - V_{RL}}{e} \tag{3.8}$$

Où e est l'entraxe du véhicule (la distance entre les deux roues arrière).

Le modèle cinématique de type char utilise seulement les sorties des encodeurs arrière du véhicule, mais il peut utiliser un gyroscope pour la mesure de la vitesse angulaire à la place du calcul basé sur l'entraxe et les vitesses des roues.

3.4 Description des filtres de Kalman

Nous avons introduit dans les sections précédentes les différents capteurs utilisés dans notre application et leurs caractéristiques, ainsi que les modèles de véhicule sur lesquels nous allons nous baser pour établir les équations cinématiques de nos filtres de Kalman. Cette section détaille les deux filtres parallèles implémentés au sein de chacune des branches de l'architecture 3.2 pour avoir un double système de localisation, et établir ainsi une redondance utilisant des capteurs indépendants permettant d'assurer des services de tolérance aux fautes. Ces services seront présentés dans la section 3.5.

3.4.1 Premier filtre de Kalman : Couplage INS - Système d'odométrie avant

Le filtre de Kalman appliqué dans la première branche de notre architecture (KF_1) consiste à coupler un système inertiel et un système d'odométrie de la roue avant du véhicule comme montré dans la figure 3.5. Nous utilisons les données d'un système inertiel INS (Accéléromètre, Gyromètre) pour prédire l'état du système. Cet état sera mis à jour et corrigé par les données du système d'odométrie de la roue avant noté *F-odo*. Ce système odométrique estime la position en employant le modèle cinématique tricycle décrit au paragraphe 3.3.1.2 ; il utilise la vitesse de la roue avant du véhicule V_F fournie par F-WSS et l'angle de braquage ϕ délivré par le capteur dédié. Les variables d'état sont les positions du véhicule x, y, son orientation θ et sa vitesse longitudinale V. Les détails de mise en œuvre de cette configuration sont les suivants :

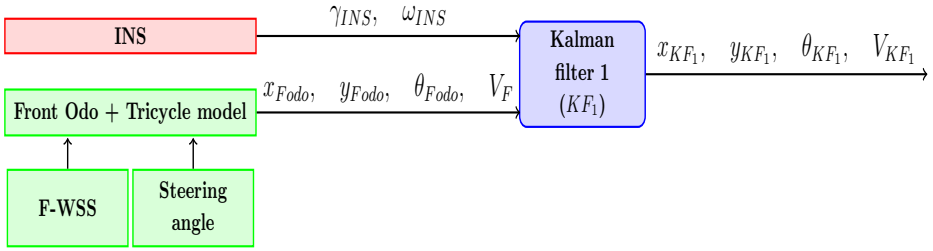

Figure 3.5 – Fusion de données d'un système inertiel et d'un système d'odométrie avant par filtre de Kalman

- *Modèle du système* : nous utilisons l'équation (3.9) comme un modèle de système dans KF_1, prenant en entrée $U_{INS}(\gamma_{INS}, \omega_{INS})$, l'accélération longitudinale γ_{INS} mesurée par l'accéléromètre et la vitesse angulaire ω_{INS} mesurée par le gyromètre.

$$\underbrace{\begin{bmatrix} x_{KF_{1_k}} \\ y_{KF_{1_k}} \\ \theta_{KF_{1_k}} \\ V_{KF_{1_k}} \end{bmatrix}}_{X_{KF_{1_k}}} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & \Delta t_k cos(\theta_{KF_{1_{k-1}}}) \\ 0 & 1 & 0 & \Delta t_k sin(\theta_{KF_{1_{k-1}}}) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{A} \cdot \underbrace{\begin{bmatrix} x_{KF_{1_{k-1}}} \\ y_{KF_{1_{k-1}}} \\ \theta_{KF_{1_{k-1}}} \\ V_{KF_{1_{k-1}}} \end{bmatrix}}_{X_{KF_{1_{k-1}}}} + \underbrace{\begin{bmatrix} \frac{1}{2}\Delta t_k^2 & 0 \\ \frac{1}{2}\Delta t_k^2 & 0 \\ 0 & 1 \\ \Delta t_k & 0 \end{bmatrix}}_{B} \cdot \underbrace{\begin{bmatrix} \gamma_{INS_{k-1}} \\ \omega_{INS_{k-1}} \end{bmatrix}}_{U_{INS_{k-1}}} + w_{KF_{1_k}} \tag{3.9}$$

La matrice de covariance de bruit du système $w_{KF_{1_k}}$ est Q_{KF_1} avec la forme suivante :

$$Q_{KF_1} = \begin{bmatrix} q_{1_x} & 0 & 0 & 0 \\ 0 & q_{1_y} & 0 & 0 \\ 0 & 0 & q_{1_\theta} & 0 \\ 0 & 0 & 0 & q_{1_V} \end{bmatrix} \tag{3.10}$$

Avec : $q_{1_x}, q_{1_y}, q_{1_\theta}, q_{1_V}$ les covariances du modèle 3.9.

- *Modèle d'observation* : Le modèle d'observation est basé sur les capteurs odométriques des roues avant du véhicule et un modèle de type tricycle [De Luca et al., 1998]. Il est mis en œuvre de la manière suivante (équation 3.11).

$$Z_{KF_{1_k}} = \begin{bmatrix} x_{Fodo} \\ y_{Fodo} \\ \theta_{Fodo} \\ V_F \end{bmatrix}_k = H_{KF_1}.X_{KF_{1_k}} + v_{KF_{1_k}} \tag{3.11}$$

$(x_{Fodo}, y_{Fodo}, \theta_{Fodo})$ est la pose du véhicule estimée par le système d'odométrie avant, et V_F est la vitesse longitudinale fournie par l'encodeur avant. La matrice d'observation H_{KF_1} utilisée dans le modèle d'observation est la matrice d'identité parce que les mesures sont les mêmes que les variables d'état du système.

La matrice de covariance de bruit de mesure $v_{KF_{1_k}}$ est R_{KF_1} définie comme suit :

$$R_{KF_1} = \begin{bmatrix} \sigma^2_{x(Fodo)} & 0 & 0 & 0 \\ 0 & \sigma^2_{y(Fodo)} & 0 & 0 \\ 0 & 0 & \sigma^2_{\theta(Fodo)} & 0 \\ 0 & 0 & 0 & \sigma^2_{V_F} \end{bmatrix} \tag{3.12}$$

avec $\sigma_{x(Fodo)}$, $\sigma_{y(Fodo)}$, $\sigma_{\theta(Fodo)}$ et σ_{V_F} les covariances du système odométrique avant $(F\text{-}odo)$.

3.4.2 Second filtre de Kalman : Couplage Système d'odomètrie arrière - GPS

Le filtre de Kalman utilisé dans la seconde branche de notre application consiste à coupler les capteurs odométriques des roues arrière du véhicule (R-WSS) avec un capteur GPS comme le montre la Figure 3.6. Les détails d'implémentation de ce filtre sont décrits ci-dessous :

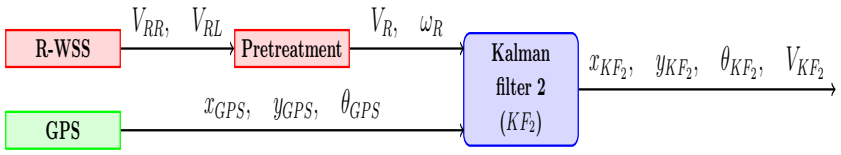

Figure 3.6 – Fusion de données d'un système d'odométrie arrière et un GPS par filtre de Kalman

- *Modèle du système* : le modèle du système est décrit dans les équations (3.13) :

$$\underbrace{\begin{bmatrix} x_{KF_{2_k}} \\ y_{KF_{2_k}} \\ \theta_{KF_{2_k}} \\ V_{KF_{2_k}} \end{bmatrix}}_{X_{KF_{2_k}}} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}}_{A} \cdot \underbrace{\begin{bmatrix} x_{KF_{2_{k-1}}} \\ y_{KF_{2_{k-1}}} \\ \theta_{KF_{2_{k-1}}} \\ V_{KF_{2_{k-1}}} \end{bmatrix}}_{X_{KF2_{k-1}}} + \underbrace{\begin{bmatrix} \Delta t_k cos(\theta_k) & 0 \\ \Delta t_k sin(\theta_k) & 0 \\ 0 & \Delta t_k \\ 1 & 0 \end{bmatrix}}_{B} \cdot \underbrace{\begin{bmatrix} V_{R_{k-1}} \\ \omega_{R_{k-1}} \end{bmatrix}}_{U_{R_{k-1}}} + w_{KF_{2_k}} \tag{3.13}$$

V_R et ω_R sont respectivement la vitesse linéaire et vitesse angulaire du véhicule. Ces valeurs sont calculées dans le bloc de *pré-traitement* de la Figure (3.6) à partir des sorties des encodeurs des roues arrière :

◇ V_R, décrit déjà auparavant, est la moyenne des vitesses droite et gauche des roues arrière.

$$V_R = \frac{V_{RR} + V_{RL}}{2} \tag{3.14}$$

◇ ω_R est la rotation élémentaire des deux roues arrière

$$\omega_R = \frac{V_{RR} - V_{RL}}{e} \tag{3.15}$$

où V_{RR} et V_{RL} sont respectivement la vitesse de la roue arrière droite et la vitesse de la roue arrière gauche, et e est l'entraxe du véhicule.

La matrice de covariance de bruit du système $w_{KF_{2_k}}$ est Q_{KF_2} :

$$Q_{KF_2} = \begin{bmatrix} q_{2_x} & 0 & 0 & 0 \\ 0 & q_{2_y} & 0 & 0 \\ 0 & 0 & q_{2_\theta} & 0 \\ 0 & 0 & 0 & q_{2_V} \end{bmatrix} \tag{3.16}$$

$q_{2_x}, q_{2_y}, q_{2_\theta}, q_{2_V}$ sont les covariances du modèle 3.13.

- *Modèle d'observation* : Les mesures GPS (x_{GPS}, y_{GPS}) sont liées à l'état du système par le modèle d'observation 3.17.

$$Z_{KF_{2_k}} = \begin{bmatrix} x_{GPS} \\ y_{GPS} \end{bmatrix}_k = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}}_{H_{KF_2}} . X_{KF2_k} + v_{KF_{2_k}} \tag{3.17}$$

La matrice de covariance du bruit de mesure GPS $v_{KF_{2_k}}$ est R_{KF_2} définie comme suit :

$$R_{KF_2} = \begin{bmatrix} \sigma^2_{x(GPS)} & 0 \\ 0 & \sigma^2_{y(GPS)} \end{bmatrix} \tag{3.18}$$

où $\sigma_{x(GPS)}, \sigma_{y(GPS)}$ sont les covariances d'observation GPS.

3.5 Services de tolérance aux fautes

Dans cette section, nous détaillons les services de détection de fautes et de rétablissement du système offerts par notre architecture dans le cadre de l'application de localisation des robots mobiles exposée précédemment (Figure 3.2).

On voit bien que notre architecture implémente deux systèmes de localisation indépendants (deux filtres de Kalman) offrant ainsi une redondance logicielle et matérielle. Le premier filtre combine les informations provenant d'une centrale inertielle (Acc, Gyro) avec celles provenant d'un système d'odométrie avant (*F-odo*) qui utilise les informations des capteurs odométriques avant du robot (F-WSS). Le second filtre fusionne les sorties des capteurs odométriques arrière du robot (R-WSS) avec les informations provenant d'un GPS.

Notre but est d'exploiter cette redondance matérielle dans les capteurs, et la diversification logicielle dans les algorithmes de filtrage de Kalman pour offrir des services de tolérance aux fautes employant la technique de comparaison. Sur la Figure 3.2, la détection de fautes est ainsi assurée par le module *MDE (Module de Détection d'Erreur)*, alors que le diagnostic et le recouvrement sont assurés par le module *MIRE (Module d'Identification et de Recouvrement d'Erreur)*. Dans ce qui suit, nous détaillons ces services de détection et de rétablissement en explicitant le fonctionnement de ces modules.

3.5.1 Service de détection d'erreurs

Pour détecter d'éventuelles erreurs dans le système, nous comparons les positions estimées par les deux filtres de Kalman implémentés. Pour cela nous calculons la distance euclidienne $\Delta Pos_{(KF_1,KF_2)}$ (équation 3.19) entre la position Pos_{KF_1} sortie du premier filtre et la position Pos_{KF_2} sortie du second filtre, et comparons ensuite cette distance à un seuil de détection spécifique $Thrs_{Det}$. Si la distance $\Delta Pos_{(KF_1,KF_2)}$ est supérieure au seuil $Thrs_{Det}$, cela implique qu'un écart significatif entre les deux positions Pos_{KF_1}, Pos_{KF_2} est survenu, que nous identifions à la présence d'une erreur. Dans le cas contraire aucune erreur n'est détectée. L'organigramme 3.7 schématise le fonctionnement de la détection de fautes.

$$\Delta Pos_{(KF_1,KF_2)} = \sqrt{(x_{KF_1} - x_{KF_2})^2 + (y_{KF_1} - y_{KF_2})^2} \tag{3.19}$$

Le seuil utilisé pour la détection d'erreur $Thrs_{det}$ est déterminé empiriquement de façon à ce que la distance $\Delta Pos_{(KF_1,KF_2)}$ ne le dépasse pas dans le comportement nominal du système (voir Chapitre 4).

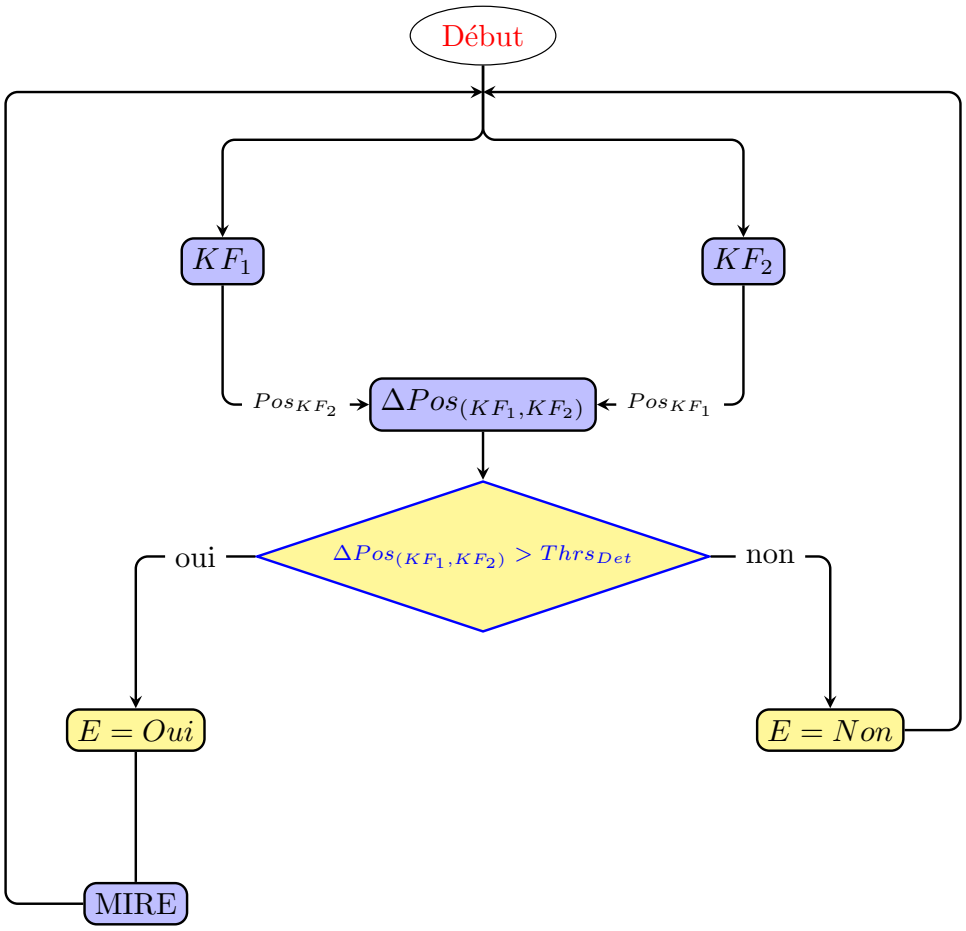

Figure 3.7 – Organigramme de détection de fautes

3.5.2 Service de diagnostic et de rétablissement du système

Quand une erreur est détectée dans le système, on procède au diagnostic de la faute qui l'a provoquée, afin d'isoler le composant fautif dans la prise de décision, et déduire par conséquent la sortie correcte du système. Ce diagnostic se fait en deux étapes successives : la comparaison des capteurs (SC) suivie ensuite par la comparaison des résidus (RC) issus des deux filtres de Kalman :

- **Comparaison des sorties des capteurs** : elle identifie, en cas de faute matérielle, le type de capteur erroné en comparant les sorties des capteurs similaires. Dans notre cas d'étude, nous comparons d'une part la sortie des capteurs odométriques arrière ($R\text{-}WSS$) avec celle de la centrale inertielle (Acc, Gyro), ces deux capteurs étant utilisés dans l'étape de prédiction de nos filtres de Kalman pour la perception de la commande du véhicule. ω_{INS} est comparée avec ω_R, et γ_{INS} comparée avec la dérivée de la vitesse linéaire issue de $R\text{-}WSS$ $\frac{\partial V_R}{\partial t}$. D'autre part la position donnée par le GPS (x_{GPS}, y_{GPS}) est comparée à celle issue du système odométrique avant (x_{Fodo}, y_{Fodo}) : ces deux blocs de

capteurs étant employés dans l'étape de correction de nos filtres de Kalman. Ces comparaisons sont réalisées par la méthode de seuillage, et trois seuils sont donc nécessaires ($Thrs_{\omega_{INS,RWSS}}$, $Thrs_{\gamma_{INS,RWSS}}$, $Thrs_{Pos_{GPS,Fodo}}$).

- ◇ Si la sortie d'un capteur s'écarte significativement de son dual, le système diagnostique une faute matérielle sur l'un de ces deux capteurs. Le capteur défectueux sera alors identifié dans la comparaison des résidus. La comparaison des sorties des capteurs se fait en calculant les trois grandeurs des equations (3.20) et en les comparant aux seuils prédéfinis ($Thrs_{\omega_{INS,RWSS}}$, $Thrs_{\gamma_{INS,RWSS}}$, $Thrs_{Pos_{GPS,Fodo}}$).
  - ✓ $Thrs_{\omega_{INS,RWSS}}$: seuil d'écart entre la vitesse angulaire donnée par le gyro ω_{INS} avec celle déduite des capteurs odométriques arrière (R-WSS) ω_R
  - ✓ $Thrs_{\gamma_{INS,RWSS}}$: seuil d'écart entre l'accélération longitudinale fournie par l'accéléromètre γ_{INS} avec la dérivée de la vitesse linéaire calculée à partir des encodeurs arrière (R-WSS) $\frac{\partial V_R}{\partial t}$
  - ✓ $Thrs_{Pos_{GPS,Fodo}}$: seuil d'écart entre la position fournie par le GPS $Pos_{GPS} = (x_{GPS}, y_{GPS})$ avec la position estimée par le système odométrique avant $Pos_{Fodo} = (x_{Fodo}, y_{Fodo})$.

$$\begin{cases} \Delta Pos_{(GPS,F-odo)} = \sqrt{(x_{GPS} - x_{Fodo})^2 + (y_{GPS} - y_{Fodo})^2} \\ \Delta\gamma_{(INS,R-WSS)} = \mid \gamma_{INS} - \frac{\partial V_R}{\partial t} \mid \\ \Delta\omega_{(INS,R_WSS)} = \mid \omega_{INS} - \omega_R \mid \end{cases} \tag{3.20}$$

- ◇ Si les sorties des capteurs sont similaires (i.e. les grandeurs calculées dans les équations 3.20 sont inférieures aux seuils choisis), le système diagnostique une faute logicielle. En l'absence de plus de redondance, il n'est pas possible de pousser plus loin le diagnostic (identifier le filtre fautif) mais seulement de mettre le système erroné dans un état sûr. Cependant un autre filtre de Kalman KF_3 à l'aide de deux sorties diversifiées parmi les capteurs utilisés pourrait être mis en œuvre pour diagnostiquer le filtre défectueux, et rétablir le système en utilisant la méthode du vote majoritaire.

Le chapitre 4 détaille comment déterminer les différents seuils de diagnostic.

- **Comparaison des résidus (CR)** : dans le cas d'une faute matérielle cette étape identifie la branche erronée d'après la valeur des résidus des filtres de

Kalman. En effet, ce résidu correspond au degré de cohérence entre les données fusionnées, une valeur haute indiquant que les données des capteurs fusionnés sont en conflit. Nous nous appuyons ici sur le mécanisme de fusion de données auquel nous ne faisons pas confiance autrement, car nous avons déjà vérifié dans l'étape *Comparaison des sorties des capteurs* que l'écart constaté entre les deux branches est dû aux capteurs et non aux mécanismes de fusion, et que nous considérons comme très faible la probabilité que deux fautes s'activent exactement au même moment.

Connaissant maintenant le type de capteur défectueux et la branche erronée, le système a identifié le capteur erroné, et peut ainsi être rétabli à l'aide des valeurs de position de la branche sans erreur.

3.5.3 Synthèse des décisions du système

La décision sur la sortie du système dans notre architecture est prise comme indiqué ci-dessous :

- Si une erreur détectée est due à une faute matérielle dans la première branche (S_1, S_2, KF_1), alors la sortie du système sera le résultat de la seconde branche Pos_{KF_2},
- Si une erreur détectée est due à une faute matérielle dans la seconde branche (S_3, S_4, KF_2), alors la sortie du système sera le résultat de la première branche Pos_{KF_1},
- Si une erreur due à une faute logicielle a été détectée, alors le système génère une alerte indiquant que la localisation est erronée,
- Si aucune erreur n'est détectée, alors la sortie du système sera la moyenne des résultats des deux branches.

3.6 Conclusion

Nous avons présenté dans ce chapitre un exemple d'application implémentant l'architecture de *duplication-comparaison* présentée dans le chapitre 2. Cette implémentation a pour objectif d'assurer la localisation d'un véhicule dans son environnement. Les mécanismes de fusion de données employés sont les filtres de Kalman utilisant des capteurs redondants et diversifiés. Nous avons montré l'applicabilité de l'architecture de duplication-comparaison, et nous avons détaillé

les services de détection et de recouvrement de fautes adaptés a la fusion de données par filtrage de Kalman.

Les mécanismes de tolérance aux fautes proposés reposent principalement sur la diversification des algorithmes de fusion de données et la réplication des capteurs de perception. La technique de seuillage et de comparaison est utilisée pour la détection et le diagnostic, et la technique de reconfiguration est utilisée pour assurer le recouvrement du système. Ces techniques sont connues et validées dans la sûreté de fonctionnement informatique, bien que la détermination des valeurs de seuils reste une difficulté connue dans la pratique.

La multiplication des capteurs de perception, et des algorithmes de fusion de données employés dans notre architecture génère un surcoût de développement et en termes financiers. Il est donc important de mesurer l'efficacité des mécanismes proposés. C'est l'objectif du prochain chapitre de ce mémoire, qui évalue les performances de notre implémentation en termes de sûreté de fonctionnement en utilisant la technique d'injection de fautes. Des fautes physiques sont injectées au niveau des capteurs de perception en altérant les sorties des capteurs, et des fautes logicielles sont injectées au niveau des deux filtres en utilisant l'injection par mutation [1] sur les modèles du système et d'observation des deux filtres de Kalman.

[1] Une mutation est une modification syntaxique élémentaire dans un code existant.

Évaluation des mécanismes de tolérance aux fautes

Sommaire

4.1 Introduction

Nous avons proposé dans le chapitre précédent plusieurs mécanismes susceptibles d'assurer la tolérance aux fautes en fusion de données par filtrage de Kalman. Nous avons présenté deux branches parallèles mettant en œuvre deux filtres de Kalman, et décrit leurs implantations pour la localisation des robots mobiles. Nous avons détaillé comment détecter des fautes et rétablir le système.

Dans ce chapitre, nous proposons une étude cherchant à évaluer cette implémentation en termes de sûreté de fonctionnement, et à montrer l'efficacité de ces mécanismes de tolérance aux fautes confrontés à des fautes. Nous présentons tout d'abord l'environnement d'expérimentation, et les principes que nous avons utilisés pour réaliser les expériences nécessaires à cette évaluation. Puis nous présentons les résultats de notre campagne d'injection de fautes.

4.2 Environnement logiciel

Notre environnement d'évaluation repose sur l'acquisition de données réelles au moyen de la plate forme logicielle PACPUS [1], le rejeu de ces données ainsi que l'injection de fautes. Le logiciel Matlab est utilisé pour ce rejeu de données et l'injection de fautes.

L'injection de fautes est nécessaire pour simuler des fautes réelles auxquelles peut être confronté le robot dans son environnement. Les fautes matérielles peuvent êtres simulées par l'altération et la modification des valeurs de sortie des capteurs. Pour les fautes logicielles, nous n'avons pas trouvé d'études explicitant quelles fautes logicielles peuvent survenir dans les mécanismes de fusion de données. Mais comme souligné dans [Lussier, 2007], des travaux existants sur les langages impératifs [Daran, 1996] et les systèmes d'exploitation [Jarbaoui, 2003] ont montré que des mutations [2] peuvent simuler efficacement des fautes logicielles réelles.

Dans notre campagne d'expérimentation les données acquises lors d'une expérience réelle sont utilisées hors ligne pour valider nos mécanismes de tolérance aux fautes présentés pour trois raisons :

- Les données sont estampées, ce qui permet de rééxecuter hors ligne les expériences, et simuler correctement le scénario exécuté par le robot dans son environnement réel. Comme le mécanisme ciblé est un mécanisme de perception dont les actions n'ont pas d'influence sur l'environnement, ce rejeu est suffisant pour caractériser son comportement.

- Pour réaliser une évaluation significative, un grand nombre d'expériences devient indispensable. Cependant les expériences réalisées directement sur des systèmes réels sont coûteuses en termes de temps d'exécution et de matériel demandé, et elles sont plus difficiles, voire impossibles, à automatiser. Par conséquent, le rejeu de données nous paraît particulièrement utile pour automatiser des expériences.

- Sur une expérience réelle, le nombre de situations adverses est limité à celles rencontrées lors de cette expérience, mais la combinaison de la méthode de rejeu de données et d'injections de fautes donne lieu à un grand nombre de situations simulées possibles.

[1] PACPUS : Perception et Assistance pour une Conduite Plus Sure
La plateforme PACPUS du laboratoire Heudiasyc a pour vocation de fédérer des outils et ressources pour mener à bien des recherches et des expérimentations dans le domaine des véhicules intelligents. L'une des principales finalités de la plate-forme consiste à développer, intégrer et tester des aides à la conduite automobile et des fonctions ADAS (Advanced Driver Assistance Systems) et de navigation autonomes.

[2] Une mutation est une modification syntaxique élémentaire dans un code existant.

Notre environnement d'évaluation est moins complexe qu'un environnement réel, mais il reflète bien la réalité, étant donné que les données utilisées dans notre expérimentation sont acquises lors d'une conduite réelle sur le terrain. Une exécution plus réaliste du robot dans son environnement, bien que préférable en termes de représentativité ou de diversification des situations adverses, aurait été bien plus difficile à mettre en pratique pour réaliser les 110 expériences que nous proposons. De plus nous souhaitons juste évaluer les performances de la perception plutôt que le système robotique dans son ensemble, et nous estimons que le rejeu de données acquises dans un environnement réel, et leur utilisation offline reste compatible avec l'évaluation des mécanismes de tolérance aux fautes.

4.2.1 Plate forme logiciel Pacpus : acquisition de données

La plateforme technologique de recherche PACPUS [3] est composée de moyens logiciels et matériels (véhicules expérimentaux instrumentés dont un exemple est détaillé précédemment Figure 3.1) développée au laboratoire Heudiasyc pour répondre aux besoins de prototypage rapide et mener des recherches et des expérimentations dans le domaine des véhicules intelligents. Pacpus [Moras, 2013] contient un ensemble d'outils logiciels, librairies et exécutables, qui permettent la création, la gestion et la communication de plusieurs composants indépendants au sein d'une application temps réel. Elle permet notamment la récupération des données capteurs en temps réel, leur enregistrement et leur datation pour un rejeu ultérieur. Pacpus est développée sur les architectures X86 et X86-64, est compatible avec plusieurs systèmes d'exploitation (Windows, Mac OS, Linux) et possède une licence Open source.

Dans notre application nous avons utilisé le véhicule Carmen sur le circuit de test représenté en Figure 4.1 et nous avons utilisé l'intergiciel (le middleware Pacpus) pour réaliser une acquisition de données datées. Pour chaque capteur utilisé, un composant récupère les données au moyen d'un thread qui scrute l'arrivée de nouvelles données et les sauvegarde dans des fichiers *".dbt"*.

4.2.2 Plate forme Matlab : Rejeu de données et Injection de fautes

Dans notre expérimentation nous nous sommes basés sur la méthode de rejeu de données. Cette technique permet de relire les données d'un fichier en temps réel. Les fichiers *".dbt"* sauvegardés lors de notre expérience réelle sur le circuit sont lus et

[3] http ://www.hds.utc.fr/pacpus

exportés dans des matrices Matlab.

Pour évaluer les mécanismes proposés, nous avons utilisé la technique d'injection de fautes qui est la méthode la plus employée pour la validation de la tolérance aux fautes. L'injection de fautes physiques est directement réalisée sur les sorties des capteurs, et l'injection de fautes logicielles est réalisée sur les modèles de processus et d'observation dans les filtres de Kalman sous Matlab. Une nouvelle sauvegarde de ces données altérées et perturbées par les fautes injectées est réalisée dans des fichiers avec les résultats de l'expérience, un répertoire spécifique étant créé pour chaque type de faute injectée.

4.3 Campagne d'injection de fautes

Dans cette section nous présentons les quatre paramètres de la campagne d'injection de fautes proposée :

- *l'activité* : la mission exécutée par le robot et son environnement.
- *les fautes* : l'ensemble des fautes matérielles et logicielles pouvant perturber le système de perception pendant l'activité du robot.
- *les relevés* : l'ensemble des informations générées lors d'une expérience.
- *les mesures* : les différents résultats significatifs représentatifs de l'efficacité du système, valables pour un ensemble d'expériences.

4.3.1 Activité

Nous appelons activité la mission demandée au robot dans un environnement donné. L'activité dans notre cas est le parcours de circuit de test montré sur la Figure 4.1. Ce circuit d'expérimentation est composé d'une ligne droite suivie d'un rond point, et puis d'une seconde ligne droite suivie d'un autre rond point pour revenir au point de départ.

Ce circuit est bien représentatif, car il simule deux principales actions du robot :

- une accélération avec une vitesse angulaire nulle pour parcourir les deux lignes droites.
- une décélération avec une vitesse angulaire non nulle pour suivre les ronds points.

Figure 4.1 – Circuit d'expérimentation

Nous avons choisi ce circuit car il donne un bon exemple d'application pour montrer les différentes situations auxquelles peut être confronté un véhicule dans son environnement, et simule les différentes manœuvres qui peuvent êtres demandées au robot à savoir le parcours d'une ligne droite, et des virages sur les ronds points.

4.3.2 Ensemble de fautes

Afin d'évaluer la performance et l'efficacité des mécanismes de détection et de rétablissement proposés, nous injectons des fautes matérielles par altération des sorties des capteurs de perception, et des fautes logicielles par mutation directement dans les modèles de processus et d'observation des filtres de Kalman. Elles sont introduites volontairement dans le code pour perturber le fonctionnement de l'architecture à des instants bien précis. Dans ce qui suit nous présentons les fautes matérielles et logicielles que nous avons introduites dans notre campagne d'expérimentation.

4.3.2.1 Fautes matérielles

Nous avons classifié les fautes matérielles que nous considérons en trois catégories différentes : *les fautes de biais*, *les fautes de trame bloquée*, et *les fautes de mise à*

zéro. Ces fautes sont détaillées dans les paragraphes suivants :

Posons S_{iout} la sortie du capteur utilisée par notre mécanisme de perception, et S_{in} sa sortie normale quand le capteur est fonctionnel c'est à dire avant l'injection.

- pour les *fautes de biais*, la sortie de capteur est biaisée par une constante Δ.

$$\underset{t_i \leq t_k}{S_{iout}(t_k)} = S_{in}(t_k) + \Delta \tag{4.1}$$

  Nous avons simulé des biais différents, que nous avons ajoutés aux sorties des capteurs à des instants distincts afin de révéler l'influence des biais sur l'estimation donnée par les filtres de Kalman, et étudier le comportement du système en réaction à ces différents biais.

- pour les *fautes de trame bloquée*, le capteur envoie toujours la même sortie à partir d'une date précise t_i.

$$\underset{t_i \leq t_k}{S_{iout}(t_k)} = S_{in}(t_i) \tag{4.2}$$

  Cette faute simule le blocage d'un capteur. Ce type de faute est fréquent dans les appareils de mesure : suite à un incident quelconque, un capteur peut se bloquer sur sa dernière sortie à tout instant dans son cycle de vie. Un exemple typique de cette faute est le blocage d'une caméra sur la dernière image acquise.

- pour les *fautes de mise à zéro*, le capteur délivre toujours la même sortie (que nous considérons ici zéro).

$$\underset{t_i \leq t_k}{S_{iout}(t_k)} = 0 \tag{4.3}$$

  Cette faute ne diffère pas beaucoup de la faute *trame-bloquée*, mais cette fois ci au lieu que le capteur se bloque sur sa dernière mesure, il se bloque sur une sortie zéro. Ce type de faute est bien représentatif du patinage ou du glissement d'un robot mobile pendant sa mission.

4.3.2.2 Fautes logicielles

Pour les fautes logicielles notre campagne d'injection de fautes utilise la technique de mutation c'est-à-dire une modification élémentaire du code. Cette mutation peut être faite soit sur le modèle de processus ou sur le modèle d'observation des deux filtres de Kalman. La technique de mutation est couramment utilisée pour les programmes impératifs, mais moins d'exemples existent pour les programmes déclaratifs. Après analyse de ces deux modèles, nous avons identifié plusieurs types de mutations possibles décrites ci dessous :

- *substitution de valeurs numériques* : chaque valeur numérique peut être remplacée par un élément d'un ensemble de valeurs numériques, comprenant un ensemble de valeurs particulières des nombres réels (telles que 0, 1 et -1). Ce premier type de fautes peut être injecté dans la matrice d'observation H des deux filtres : en substituant ses éléments, un 0 devient un 1 ou -1 ou vice-versa.

- *substitution d'expression géométrique* : Chaque variable exprimant une fonction géométrique peut être remplacée par une autre expression géométrique (un *cos* devient un *sin* par exemple...). Ces fautes sont injectées directement dans les matrices de transition A qui permettent de lier deux états successifs (k, et $k+1$). Cette substitution est faite sur les modèles de processus des deux filtres de Kalman.

- *substitution de signe d'une variable* : Cet exemple de faute de programmation est typique dans le développement de logiciels. cette mutation consiste a mettre un - à la place d'un $+$, et vice-versa, et * à la place de $/$, etc.

- *mauvaise estimation des matrices Q, R* : étant donné que les matrices Q, R de (bruits de modèle w et de mesure v respectivement) sont initialisées empiriquement par l'expérimentation, une erreur dans l'estimation de leurs valeurs est hautement possible et peut avoir des conséquences sur la sortie du filtre de Kalman. Dans notre campagne d'injection de fautes nous avons simulé certaines fautes de ce type afin de révéler l'effet de mauvaise estimation des matrices de covariance Q et R sur la sortie du système.

On peut énumérer un nombre important de mutations, mais dans notre expérimentation nous nous sommes limités à ces quatre types de fautes.

Pour simplifier la lecture et la compréhension des différentes mutations réalisées dans notre campagne d'injection de fautes, nous avons nommé les différents éléments des matrices de transition A, de commande B, et d'observation H employées dans nos filtres de Kalman selon les notations montrées dans la table 4.1. Les 20 mutations citées que nous avons réalisées sont représentées dans la dernière colonne de cette table. Les dix premières lignes représentent les mutations réalisées sur le filtre de Kalman KF_1, et les dix dernières sur le second filtre KF_2. Par exemple, la première mutation modélise une mauvaise estimation des matrices de covariance Q et R, alors que la seconde mutation substitue la variable géométrique "cos" dans le modèle de filtre KF_1 par un "sin".

Bloc	Modèle	Éléments matriciel	Symboles	Mutation	
				Original	Mutant
KF_1	Modèle du système	$\underbrace{\begin{bmatrix} 1 & 0 & 0 & \Delta t_k cos(\theta_{KF_{1_{k-1}}}) \\ 0 & 1 & 0 & \Delta t_k sin(\theta_{KF_{1_{k-1}}}) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{A_{KF_1}}$	$\underbrace{\begin{bmatrix} A^1_{1,1} & A^1_{1,2} & A^1_{1,3} & A^1_{1,4} \\ A^1_{2,1} & A^1_{2,2} & A^1_{2,3} & A^1_{2,4} \\ A^1_{3,1} & A^1_{3,2} & A^1_{3,3} & A^1_{3,4} \\ A^1_{4,1} & A^1_{4,2} & A^1_{4,3} & A^1_{4,4} \end{bmatrix}}_{A_{KF_1}}$	$A^1_{1,4} : cos(\theta)$ $A^1_{2,4} : sin(\theta)$ $A^1_{1,4} : cos(\theta)$ $A^1_{2,4} : sin(\theta)$	$A^1_{1,4} : sin(\theta)$ $A^1_{2,4} : cos(\theta)$ $A^1_{1,4} : -cos(\theta)$ $A^1_{2,4} : -sin(\theta)$
KF_1	Modèle du système	$\underbrace{\begin{bmatrix} \frac{1}{2}\Delta t_k^2 & 0 \\ \frac{1}{2}\Delta t_k^2 & 0 \\ 0 & 1 \\ \Delta t_k & 0 \end{bmatrix}}_{B_{KF_1}}$	$\underbrace{\begin{bmatrix} B^1_{1,1} & B^1_{1,2} \\ B^1_{2,1} & B^1_{2,2} \\ B^1_{3,1} & B^1_{3,2} \\ B^1_{4,1} & B^1_{4,2} \end{bmatrix}}_{B_{KF_1}}$	$B^1_{1,1} : \frac{1}{2}\Delta t^2$ $B^1_{1,1} : \frac{1}{2}\Delta t^2$	$B^1_{1,1} : \Delta t^2$ $B^1_{1,1} : \frac{1}{4}\Delta t^2$
KF_1	Modèle d'observation	$\underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{H_{KF_1}}$	$\underbrace{\begin{bmatrix} H^1_{1,1} & H^1_{1,2} & H^1_{1,3} & H^1_{1,4} \\ H^1_{2,1} & H^1_{2,2} & H^1_{2,3} & H^1_{2,4} \\ H^1_{3,1} & H^1_{3,2} & H^1_{3,3} & H^1_{3,4} \\ H^1_{4,1} & H^1_{4,2} & H^1_{4,3} & H^1_{4,4} \end{bmatrix}}_{H_{KF_1}}$	$H^1_{1,1} : 1$ $H^1_{1,2} : 0$ $H^1_{1,1} : 1$	$H^1_{1,1} : 0$ $H^1_{1,2} : 1$ $H^1_{1,1} : -1$
KF_2	Modèle du système	$\underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}}_{A_{KF_2}}$	$\underbrace{\begin{bmatrix} A^2_{1,1} & A^2_{1,2} & A^2_{1,3} & A^2_{1,4} \\ A^2_{2,1} & A^2_{2,2} & A^2_{2,3} & A^2_{2,4} \\ A^2_{3,1} & A^2_{3,2} & A^2_{3,3} & A^2_{3,4} \\ A^2_{4,1} & A^2_{4,2} & A^2_{4,3} & A^2_{4,4} \end{bmatrix}}_{A_{KF_2}}$	$A^2_{1,1} : 1$ $A^2_{1,1} : 1$ $A^2_{1,2} : 0$	$A^2_{1,1} : 0$ $A^2_{1,1} : -1$ $A^2_{1,2} : 1$
KF_2	Modèle du système	$\underbrace{\begin{bmatrix} \Delta t_k cos(\theta_k) & 0 \\ \Delta t_k sin(\theta_k) & 0 \\ 0 & \Delta t_k \\ 1 & 0 \end{bmatrix}}_{B_{KF_2}}$	$\underbrace{\begin{bmatrix} B^2_{1,1} & B^2_{1,2} \\ B^2_{2,1} & B^2_{2,2} \\ B^2_{3,1} & B^2_{3,2} \\ B^2_{4,1} & B^2_{4,2} \end{bmatrix}}_{B_{KF_2}}$	$B^2_{1,1} : \cos(\theta)$ $B^2_{1,1} : \sin(\theta)$ $B^2_{1,2} : \Delta t^2$	$B^2_{1,1} : \cos(\theta)$ $B^2_{1,1} : \cos(\theta)$ $B^2_{1,2} : \frac{1}{4}\Delta t^2$
KF_2	Modèle d'observation	$\underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}}_{H_{KF_2}}$	$\underbrace{\begin{bmatrix} H^1_{1,1} & H^1_{1,2} & H^1_{1,3} & H^1_{1,4} \\ H^1_{2,1} & H^1_{2,2} & H^1_{2,3} & H^1_{2,4} \end{bmatrix}}_{H_{KF_1}}$	$H^2_{1,1} : 1$ $H^2_{1,2} : 0$ $H^2_{1,1} : 1$	$H^2_{1,1} : 0$ $H^2_{1,2} : 1$ $H^2_{1,1} : -1$

Tableau 4.1 – Notations symboliques

4.3.3 Relevés

Plusieurs fichiers journaux intermédiaires sont produits par une expérience : les données des capteurs avec fautes injectées, les positions estimées par les filtres, les différentes grandeurs utilisées pour la détection et le rétablissement du système. Toutes ces données engendrées par une expérience sont sauvegardées dans des fichiers journaux dans des répertoires spécifiques à chaque type de faute injectée.

4.3.4 Mesures

A partir des différents relevés, nous essayons d'établir des mesures significatives représentant l'efficacité de l'architecture proposée. En effet, contrairement à des expériences d'injection de fautes sur des systèmes informatisés plus classiques, le résultat d'une expérience ne peut pas être facilement distingué par un état de réussite ou d'échec. Comme indiqué précédemment, un système de perception est confronté à des environnements ouverts et mouvants ce qui engendre un grand nombre de situations.

Pour caractériser le comportement du système développé face à une faute injectée, nous avons défini différentes mesures compréhensibles et significatives : Del_{Det}, B_d, B_i, B_r, B_{Err}, B_{Def}, P_{FP}, P_{FN}, P_i, P_r, P_{ESD}, Del_{Def}, Del_{Err}, et $Del_{Def,Det}$

- *Le délai de détection* (Del_{Det}) est la différence entre l'instant d'injection de la faute t_{Inj} et l'instant de détection t_{Det} (i.e. l'instant oú la différence entre les deux positions estimées par les deux filtres de Kalman dépasse le seuil de détection prédéfini $Thrs_{Det}$.

$$\begin{cases} Del_{Det} = t_{Det} - t_{Inj} \\ \text{Avec} \\ t_{Det} = \underset{\Delta Pos_{KF_1,KF_2} \geq Thrs_{Det}}{t_k} \end{cases} \tag{4.4}$$

  Cette mesure nous permettra d'avoir un avis sur le délai de détection, car la faute injectée doit vite être détectée pour qu'elle soit diagnostiquée et recouverte avant qu'une seconde faute soit survenue.

Pour déterminer si la faute injectée est bien détectée, diagnostiquée et identifiée, et que le système est correctement rétabli, nous avons défini trois variables booléennes pour chacune des expériences effectuées :

- B_d caractérise si la faute est détectée ($B_d = 1$ si la détection a lieu, $B_d = 0$ sinon),

- B_i représente le fait que le type de faute détectée est correctement diagnostiquée (*Matériel* dans les capteurs capteurs ou *logiciel* dans les algorithmes de fusion de données) ($B_i = 1$ si une faute est correctement identifiée comme matérielle, $B_i = 0$ sinon),

- B_r détermine si le diagnostic exact de la faute et le rétablissement du système après détection est réalisé ($B_r = 1$ si le système est correctement rétabli, $B_r = 0$ sinon).

Avant de présenter d'autres mesures, nous définissons la grandeur $diff_{CF,CN}$ qui représente l'écart entre le comportement nominal du système et son comportement avec la faute injectée. Cette grandeur est ainsi la différence entre la position nominale, et la position erronée.

Posons $(x_{KF_{1N}}, y_{KF_{1N}})$, $(x_{KF_{2N}}, y_{KF_{2N}})$ les sorties des deux filtres de Kalman KF_1, KF_2 en absence de fautes, et $(x_{KF_{1F}}, y_{KF_{1F}})$, $(x_{KF_{2F}}, y_{KF_{2F}})$ les sorties des deux filtres de Kalman KF_1, KF_2 respectivement avec une faute injectée. Du fait de l'hypothèse de faute unique, seule une branche est concernée par la faute. Dans la suite on supposera qu'il s'agit de la branche 2 (choix arbitraire puisque les deux possibilité sont symétriques). Dans ce cas $(x_{KF_{1F}}, y_{KF_{1F}}) = (x_{KF_{1N}}, y_{KF_{1N}})$.

Sur la figure 4.2, nous définissons $P_N(x_N, y_N)$ le milieu du segment constitué par les deux extrémités $((x_{KF_{1N}}, y_{KF_{1N}}), (x_{KF_{2N}}, y_{KF_{2N}}))$, et $P_F(x_F, y_F)$ le milieu du segment constitué par les deux extrémités $((x_{KF_{1F}}, y_{KF_{1F}}), (x_{KF_{2F}}, y_{KF_{2F}}))$ tels qu'ils sont définies dans l'équation 4.5.

$$\begin{cases} x_F = \frac{x_{KF_{1F}} + x_{KF_{2F}}}{2} \text{ et } y_F = \frac{y_{KF_{1F}} + y_{KF_{2F}}}{2} \\ x_N = \frac{x_{KF_{1N}} + x_{KF_{2N}}}{2} \text{ et } y_N = \frac{y_{KF_{1N}} + t_{KF_{2N}}}{2} \end{cases} \tag{4.5}$$

Nous définissons la grandeur $diff_{CF,CN}$ la distance entre les deux points P_N et P_F suivant l'équation 4.6 :

$$diff_{CF,CN} = \sqrt{(x_F - x_N)^2 + (y_F - y_N)^2} \tag{4.6}$$

Dans nos expériences d'injection de fautes, notre mécanisme de détection ne connait que $(x_{KF_{1F}}, y_{KF_{1F}})$ et $(x_{KF_{2F}}, y_{KF_{2F}})$ sorties des deux filtres de Kalman avec la faute injectée. Cependant pour évaluer la tolérance aux fautes une comparaison de ces sorties avec celles de comportement nominal est nécessaire, ce qui est possible dans notre cas puisque les fautes injectées intentionnellement. Il est évident que le système ne dispose pas de cette information et doit se reposer seulement sur les valeurs de ses deux filtres de Kalman pour détecter la faute. Pour cette raison on se place en tant qu'observateur "omniscient" et on peut donc utiliser également les

sorties des filtres dans le comportement nominal pour calculer $diff_{CF,CN}$.

Pour caractériser notre système en termes de *faux positifs* [4] et d'*absence de détection* [5] nous définissons encore deux variables booléennes B_{Err} et B_{Def}

- B_{Err} détermine si la faute injectée engendre une erreur dans le système, sans qu'il soit forcement considéré défaillant. Nous posons ici que le système contient une erreur significative si la différence entre le comportement nominal et le comportement erroné du système est supérieure à 2 mètres. Cette variable ($B_{Err} = 1$ si le système contient une erreur, $B_{Err} = 0$ sinon) est définie comme suit :

$$\begin{cases} B_{Err} = 1 \text{ Si } diff_{CF,CN} >= (Thrs_{Err} = 2m) \\ B_{Err} = 0 \text{ Si } diff_{CF,CN} < (Thrs_{Err} = 2m) \end{cases} \tag{4.7}$$

- B_{Def} détermine si la faute injectée engendre une défaillance du système. Nous considérons que le système étudié est défaillant si la différence entre le comportement nominal et le comportement avec la faute injectée est supérieure a ($Thrs_{Def} = 10m$). Cette variable ($B_{Def} = 1$ si le système est défaillant, $B_{Def} = 0$ sinon) est donc définie comme suit :

$$\begin{cases} B_{Def} = 1 \text{ Si } diff_{CF,CN} >= (Thrs_{Def} = 10m) \\ B_{Def} = 0 \text{ Si } diff_{CF,CN} < (Thrs_{Def} = 10m) \end{cases} \tag{4.8}$$

Nous avons ici défini les deux seuils $Thrs_{Err}$ et $Thrs_{Def}$ à partir du comportement nominal. Cependant, dans des applications réelles, ces deux seuils sont en général donnés par la spécification du système à développer et c'est à partir de ces valeurs que les seuils de détection ($Thrs_{Det}$), de diagnostic ($Thrs_{Pos_{GPS,F-odo}}$, $Thrs_{\gamma_{(INS,R-WSS)}}$ et $Thrs_{\omega_{(INS,R-WSS)}}$) et de rétablissement ($Thrs_{Res}$) seront définis.

Nous avons déterminé les valeurs de $Thrs_{Err}$ et $Thrs_{Def}$ en étudiant la figure 4.2, et en particulier deux de ses cas limites Figure4.3 et 4.4 à partir d'une valeur $Thrs_{Det} = 10m$.

- $Thrs_{Err}$: c'est l'écart minimum entre les comportements nominal et erroné du système pour lequel nous jugeons l'erreur comme significative. Ce seuil sera utilisé pour déterminer les *faux positifs*, et il serait donc souhaitable pour le système qu'il soit proche de la différence minimale causant la détection d'erreur. La détermination de cette valeur est présentée dans la figure 4.3. KF_{2F} est le plus proche possible de la valeur maximale expérimentale de KF_{2N}, tout en déclenchant une détection d'erreur. La valeur théorique

[4] Faux positifs : alertes en absence d'erreurs dans le système.

[5] Absence de détection : absence d'alertes en présence d'erreurs dans le système.

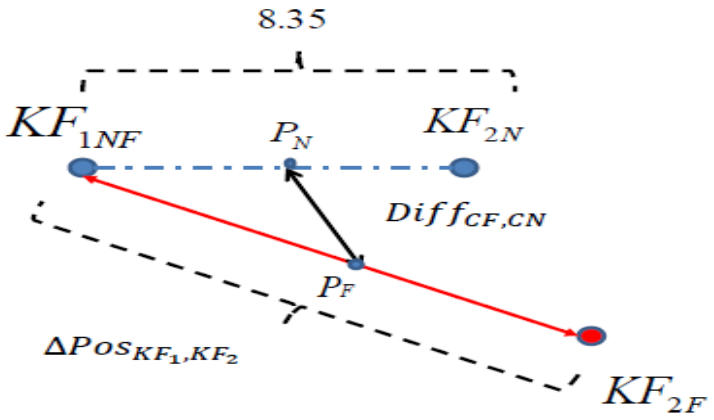

Figure 4.2 – Détermination de $Thrs_{Err}$ et $Thrs_{Def}$

maximale garantissant une absence de faux positifs est donc de 0.825 mètres. Nous avons décidé de prendre une valeur de $Thrs_{Err}$ de 2 mètre, supérieure à 0.825 mètres, ce qui pourra occasionner de faux positifs dans nos résultats.

- $Thrs_{Def}$: c'est l'écart minimum entre le comportement nominal et erroné pour lequel nous considérons le système comme défaillant. Ce seuil sera utilisé pour déterminer les *absences de détection*, et est déterminé à partir de la figure 4.4. En effet dans celle-ci KF_{2F} est le plus loin possible de la valeur maximale expérimentale de KF_{2N}, tout en ne déclenchant pas d'erreur. La valeur théorique minimale garantissant la détection est donc 9.17 mètres. Avec une valeur $Thrs_{Def}$ de 10 mètres, nous garantissons que toutes les défaillances causées par les fautes injectées seront détectées.

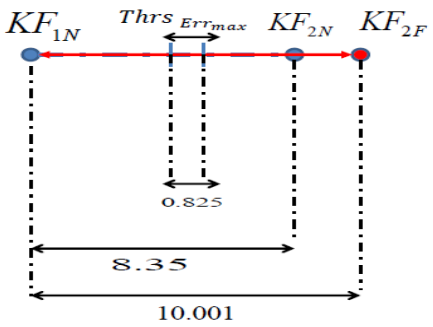

Figure 4.3 – Détermination de $Thrs_{Err}$

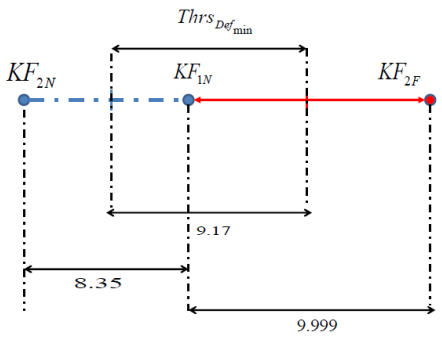

Figure 4.4 – Détermination de $Thrs_{Def}$

A partir de ces mesures, nous définissons encore pour chaque type de faute

injectée (matérielle, ou logicielle) certains pourcentages caractérisant la détection, l'identification et le recouvrement dans nos expériences :

- P_{FP} représente le taux de *faux positifs* [6] dans le système. Un *faux positif* est déclaré si la faute injectée est détectée comme erreur ($B_d = 1$) sans que le système soit erroné ($B_{Err} = 0$) :

$$\begin{cases} P_{FP} = \dfrac{\text{Nombre de faux positifs}}{\text{Nombre de fautes détectées}} \\ \text{avec :} \\ B_d \& \overline{B_{Err}} \quad \Rightarrow \text{faux positif} \end{cases} \tag{4.9}$$

  A noter que d'après la valeur choisi de $Thrs_{Err}$, de tels *faux positifs* seront possibles. La figure 4.5 montre un exemple de faux positif. En effet sur cette figure la valeur de $\Delta Pos_{KF_{1N},KF_{2F}}$ est égale à 11 mètres. Elle est strictement supérieure à la valeur du seuil de détection ($Thrs_{Det} = 10$ mètres), par conséquent notre architecture signale une détection d'erreur. Cependant que la différence entre le comportement nominal du système et son comportement avec la faute injectée c'est-à-dire la distance entre les deux points P_N et P_F ($diff_{CF,CN}$) est égale à 1.5 mètres. Elle est strictement inférieure à la valeur du seuil ($Thrs_{Err} = 2$ mètres), par conséquent notre système détecte une erreur qui n'est pas significative que nous considérons dans notre étude comme un faux positif.

- P_{ND} représente le taux d'*absence de détection* [7] dans le système. Une *absence de détection* est déclarée si l'erreur engendrée par la faute injectée n'est pas détectée ($B_d = 0$) sachant que le système est jugé défaillant ($B_{Def} = 1$) :

$$\begin{cases} P_{ND} = \dfrac{\text{Nombre de fautes non detectées}}{\text{Nombre fautes causant une defaillance}} \\ \text{avec :} \\ \overline{B_d} \& B_{Def} \quad \Rightarrow \text{absence de détection} \end{cases} \tag{4.10}$$

  A noter que selon la valeur choisie de $Thrs_{Def}$ d'après la figure 4.4, de telles absences de détection ne peuvent pas se produire.

A noter qu'il peut aussi y avoir des erreurs sans défaillance et sans détection (Figure 4.6). Dans ce cas la différence entre le comportement nominal du système et son comportement avec la faute injectée ($diff_{CF,CN}$) est égale à 4 mètres. Cette valeur est strictement supérieure au seuil ($Thrs_{Err} = 2$ mètres), ce qui signifie qu'une erreur significative est présente dans le système. Cependant la distance

[6] faux positifs : alertes en absence d'erreurs dans le système

[7] Absence de détection : absence d'alertes en présence d'erreurs dans le système

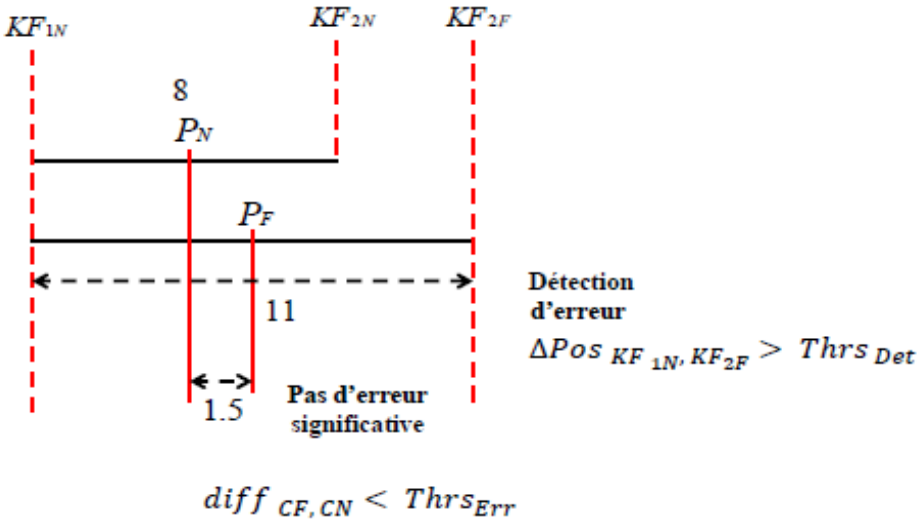

Figure 4.5 – Exemple de faux positif

$\Delta Pos_{KF_1,KF_2}$ est égale à 4 mètres. Elle est strictement inférieure au seuil de détection choisi ($Thrs_{Det} = 10$ mètres), ce qui implique qu'il n'y a pas de détection. Ce comportement est correct d'après nos définitions : bien qu'il y ait une erreur, il n' y a pas de défaillance et le système n'a pas d'obligation à soulever une détection d'erreur.

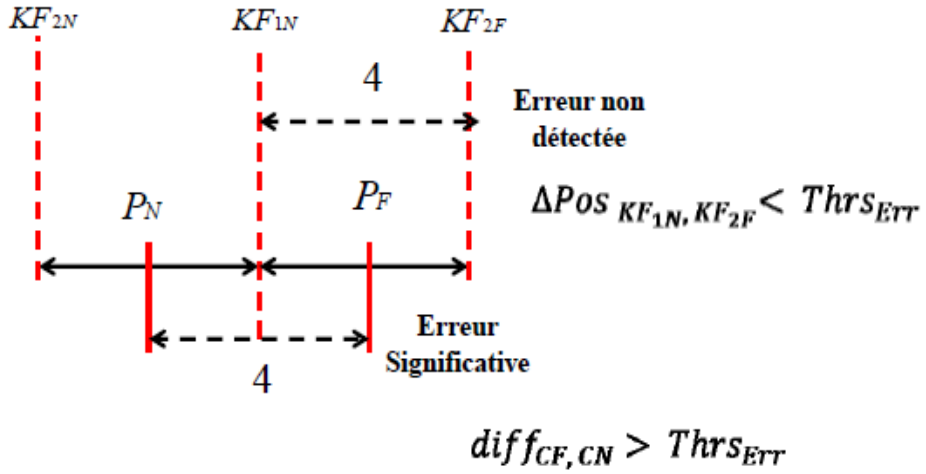

Figure 4.6 – Exemple d'erreur non détectée

- P_i est le pourcentage de fautes détectées correctement diagnostiquées et identifiées (au sens du diagnostic du paragraphe 3.5.2).

$$P_i = \frac{\text{Nombre de fautes correctement identifiées}}{\text{Nombre de fautes detectées}} \tag{4.11}$$

- P_r : est le pourcentage pour lequel le système est bien rétabli (au sens du paragraphe 3.5.2).

$$P_r = \frac{\text{Nombre de fois que le systeme est correctement rétabli}}{\text{Nombre de fautes detectées}} \tag{4.12}$$

- P_{ESD} : est le pourcentage de fautes détectées avec comportement erroné sans comportement défaillant du système : C'est le taux de fautes pour lesquelles l'expression logique $B_d \& B_{Err} \& \overline{B_{Def}}$ est vraie. Pour ces fautes la détection est correcte il y a bien une erreur mais ces erreurs n'ont pas causé de défaillance et pourraient à la rigueur être considérées comme des détections intempestives. Ce sera au concepteur de décider quel comportement du système définir face à ces fautes. A noter que des perturbations externes importantes sur les capteurs (par exemple reflet de soleil sur une camera, des saut GPS, du bruit de mesure sur les centrales inertielles) peuvent se produire et pourraient être traitées soit en relevant également le seuil de détection, soit en les considérant comme des fautes temporaires qui disparaitraient du système au même temps que la perturbation.

A noter que P_i et P_r sont indépendants des valeurs $Thrs_{Err}$ et $Thrs_{Def}$ choisies.

Dans notre expérimentation nous avons aussi défini trois variables temporelles caractérisant notre système :

- Del_{Def} détermine au bout de combien de temps la faute injectée cause une défaillance du système :

$$\begin{cases} Del_{Def} = t_{Def} - t_{Inj} \\ \text{Avec} \\ t_{Def} = \underset{Diff(CF,CN) \geq Thrs_{Def}}{t_k} \end{cases} \tag{4.13}$$

- Del_{Err} détermine au bout de combien de temps une erreur significative est détectée.

$$\begin{cases} Del_{Err} = t_{Det} - t_{Err} \\ \text{Avec} \\ t_{Err} = \underset{Diff(CF,CN) \geq Thrs_{Err}}{t_k} \end{cases} \tag{4.14}$$

- $Del_{Def,Det}$ détermine le temps qui reste après détection jusqu'à la défaillance (une défaillance étant forcément détectée vu les valeurs de seuils).

$$\begin{cases} Del_{Def,Det} = t_{Def} - t_{Det} \\ \text{Avec} \\ t_{Det} = \underset{\Delta Pos_{KF_1,KF_2} \geq Thrs_{Det}}{t_k} \end{cases} \tag{4.15}$$

  Théoriquement $Del_{Def,Det} \geq 0$ car $t_{Def} \geq t_{Det}$ du fait que d'après l'étude de la figure 4.4 il ne peut pas exister d'instant t où ($B_{Def} = 1$ et $B_d = 0$).

Les seuils sont choisis pour que :

- toute défaillance détectée soit aussi une erreur détectée,
- il n'y ait pas de défaillance non détectée. Il peut en revanche y avoir des erreurs sans défaillance et sans détection ainsi que des faux positifs et des erreurs détectées sans défaillance.

La figure 4.7 montre ainsi les différentes situations rencontrées dans notre étude expérimentale.

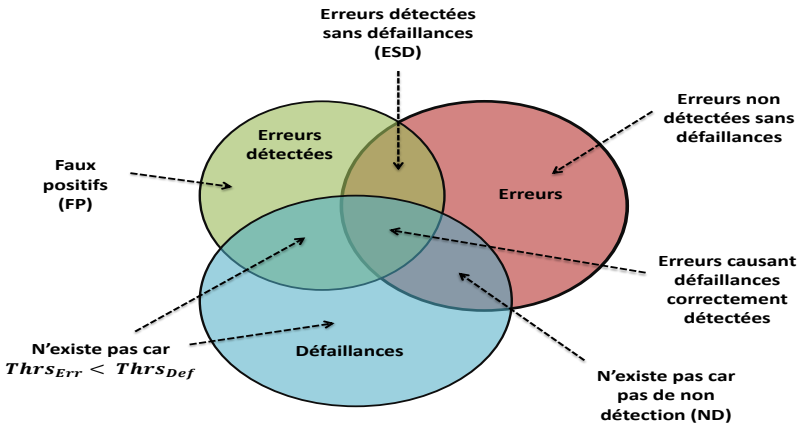

Figure 4.7 – Différentes situations de détection, d'erreur et de défaillance rencontrées dans notre campagne d'expérimentations

La table 4.2 résume les différentes mesures employées dans notre campagne d'évaluation des performances des mécanismes proposés :

Mesure	Désignation	Unité
Del_{Det}	Délai de détection	Seconde
B_d	Indicateur de détection	Booléen
B_i	Indicateur d'identification	Booléen
B_r	Indicateur de rétablissement	Booléen
B_{Err}	Indicateur d'erreur	Booléen
B_{Def}	Indicateur de défaillance	Booléen
P_{FP}	Taux de faux positifs	%
P_{ND}	Taux d'absence de détection	%
P_i	Taux d'identification	%
P_r	Taux de rétablissement	%
P_{ESD}	Taux d'erreur sans défaillance	%
Del_{Def}	Délai de défaillance	Seconde
Del_{Err}	Délai d'erreur	Seconde
$Del_{Def,Det}$	Délai entre la détection et la défaillance	Seconde

Tableau 4.2 – Résumé des mesures pour le cas d'application de localisation d'un robot mobile

4.4 Exemple d'expérimentation

Dans cette section nous présentons un échantillon de notre campagne d'expérimentation. Nous exposons les résultats de quatre types d'injection (une faute de saut sur la sortie du GPS, une faute de mise à zero sur l'angle de braquage, une faute de biais sur les encodeurs F-WSS, et une faute logicielle sur le filtre de Kalman KF_1). Nous présentons d'abord le comportement nominal du système, et ensuite les résultats de nos expériences en présence de ces fautes injectées.

4.4.1 Remarques préliminaires

Avant de présenter les résultats de nos expériences, nous faisons d'abord quelques remarques et observations liées à notre campagne d'injection de fautes :

- il est supposé que seulement une seule erreur se produit dans le système à un instant donné, que ce soit à cause d'une faute matérielle dans l'un des capteurs ou une faute logicielle dans l'un des deux filtres de Kalman (hypothèse de faute unique).
- les fautes injectées sont permanentes, ce qui veut dire qu'une fois qu'une faute est injectée sur l'un des composants du système (capteur, filtre de Kalman), elle reste présente jusqu'à la fin de la mission.
- tous les seuils utilisés pour la détection et le diagnostic sont déterminés

empiriquement et ont été choisis pour être significativement plus élevés que les valeurs observées dans le comportement nominal. C'est un problème classique en tolérance aux fautes.

- les matrices de covariances de bruit de modèle Q et de mesure R sont déterminées de telle manière à ce qu'elles reflètent l'estimation du bruit réel de chaque capteur sur le filtre de Kalman. Par définition, ce bruit dépendant des circonstances environnementales ne peut pas être connu et n'est donc qu'estimé.

4.4.2 Comportement nominal sans injection de fautes

Pour évaluer les différents mécanismes développés au cours de ces travaux, nous avons réalisé des acquisitions de données avec le véhicule instrumenté dans des conditions de circulations réelles. Le véhicule parcourt une distance de 420 mètres sur le circuit de test présenté en figure 4.1, et pour chaque capteur utilisé, un composant récupère les données au moyen d'un thread qui scrute l'arrivée de nouvelles données et les sauvegarde dans des fichiers ".dbt", les données recueillies sont :

- les données odométriques : les capteurs odométriques sont échantillonnés avec une fréquence de 25 Hz en fournissant 3224 sorties
- la position et la vitesse GPS : le GPS a une fréquence de 10 Hz en fournissant 1312 mesures
- les données inertielles : la centrale inertielle a une fréquence fondamentale de 100 Hz, en délivrant 12595 sorties

Nous avons réalisé une interpolation linaire de ces différentes mesures pour synchroniser les données à utiliser par les filtres de Kalman.

Une expérience sans faute a été réalisée pour caractériser le comportement nominal du système. Ces résultats nous sont utiles pour fixer les différents seuils de détection et de diagnostic employés dans notre architecture, et sert comme échantillon de référence pour la comparaison au comportement en présence de fautes.

La Figure 4.8 représente la position du véhicule estimée par les deux filtres de Kalman (Pos_{KF_1}, Pos_{KF_2}) et la sortie de notre architecture Pos_S, qui est la moyenne de ces deux estimations. La Figure 4.9 représente la distance $\Delta Pos_{(KF_1,KF_2)}$. Dans ce cas nominal cette distance est inférieure au seuil de détection choisi $Thrs_{Det}$ (10 m).

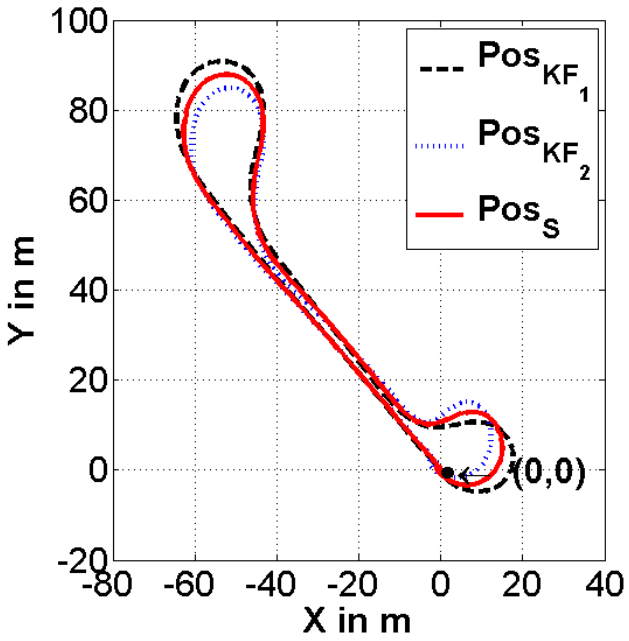

Figure 4.8 – Comportement nominal : La position estimée par les deux filtres de Kalman et le système

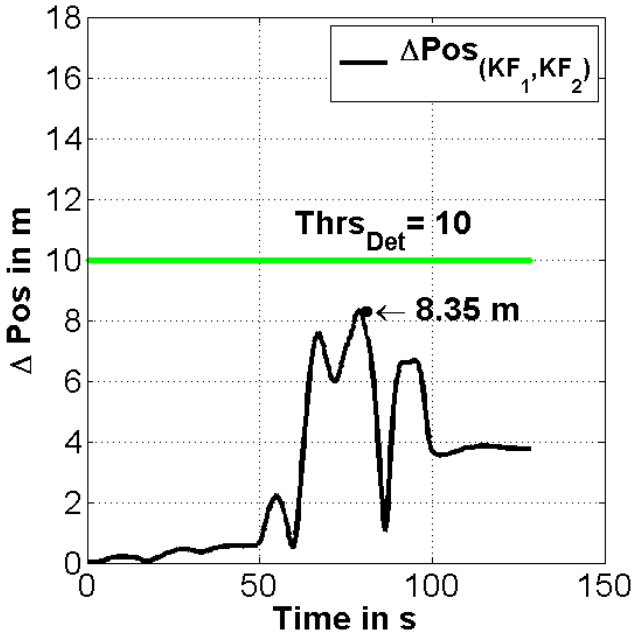

Figure 4.9 – Comportement nominal : Distance entre les positions des deux filtres de Kalman

Trois autres seuils sont nécessaires dans notre implémentation :

- Un seuil pour comparer la sortie du GPS avec celle du système d'odométrie de roues avant (F-odo) : $Thrs_{Pos_{GPS,F-odo}} = 10$ mètres.

- Deux seuils pour comparer les sorties de la centrale inertielle (INS) avec les valeurs issues des pré-traitements sur les sorties des capteurs odométriques des roues arrières (R-WSS) : $Thrs_{\gamma_{(INS,R-WSS)}} = 0.45\ m/s^2$, et $Thrs_{\omega_{(INS,R-WSS)}} = 0.23\ rad/s$.

- Un seuil pour s'assurer, en cas d'erreur matérielle détectée, que le conflit dans un filtre de Kalman est considérablement plus élevé que dans l'autre $Thrs_{Res} =$ 9 mètres.

Tous ces seuils ont été choisis pour être significativement plus élevés que les valeurs observées dans le comportement nominal. La table 4.3 résume les différents seuils employés dans notre campagne d'expérimentation.

Seuils	**Désignation**	**Valeur**	**Unité**
$Thrs_{Det}$	Seuil de détection	10	Mètres
$Thrs_{Pos_{GPS,F-odo}}$	Seuil de comparaison des sorties de GPS et du système F-odo	10	Mètres
$Thrs_{\gamma_{(INS,R-WSS)}}$	Seuil de comparaison des accélérations issues de INS et R-WSS	0.45	$Metres^2/Seconde$
$Thrs_{\omega_{(INS,R-WSS)}}$	Seuil de comparaison des vitesses angulaires issues de INS et R-WSS	0.23	Radian/Seconde
$Thrs_{Res}$	Seuil de comparaison des résidus	9	Mètres
$Thrs_{Err}$	écart minimum entre les comportements nominal et erroné du système pour lequel nous jugeons l'erreur comme significative	2	Mètres
$Thrs_{Def}$	écart minimum entre le comportement nominal et erroné pour lequel nous considérons le système comme défaillant	10	Mètres

Tableau 4.3 – Résumé des seuils pour le cas d'application de localisation d'un robot mobile

4.4.3 Comportement en présence de fautes

Nous présentons ici les résultats de quatre injections de fautes : trois fautes matérielles (l'une sur le GPS, une sur le capteur d'angle de braquage, et l'autre

sur les encodeurs F-WSS), et une faute logicielle dans le filtre de Kalman KF_1.

- **Faute de saut sur le GPS**

  La première faute injectée que nous considérons est une faute additive permanente dans la sortie du GPS. Cette faute externe typique dans la réalité simule un saut dans la position fournie par le GPS en raison de rebonds d'un des signaux satellites. Cette faute n'est généralement pas permanente, mais elle peut se produire pendant un temps significatif. A l'instant $t_i = 60.5$ secondes, nous avons ajouté un saut de 10 mètre sur la composante x_{GPS}, comme décrit dans l'équation (4.16).

$$\underset{t_i \leq t_k}{x_{GPS}(t_k)} = x_{GPS}(t_i) + 10 \tag{4.16}$$

  Dans la Figure 4.10, nous voyons que la différence entre les positions estimées par les deux filtres de Kalman $\Delta Pos_{(KF_1,KF_2)}$ dépasse le seuil $Thrs_{Det}$, donc le système détecte la présence d'une erreur à l'instant $t_{Det} = 62.5$ secondes.

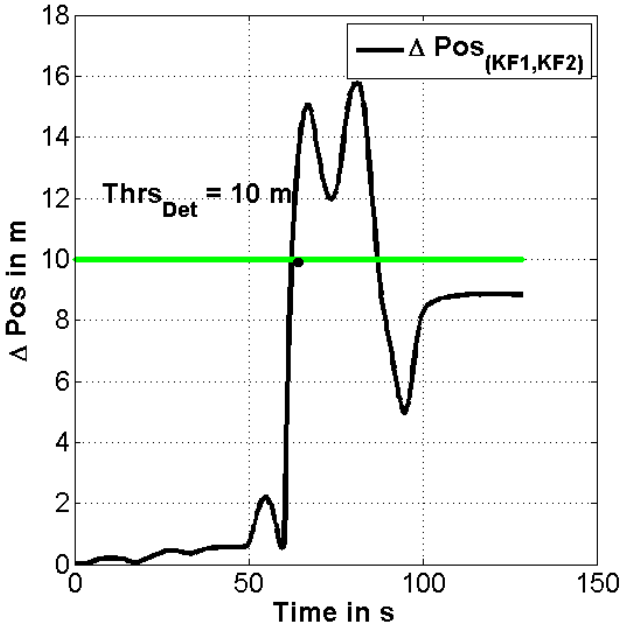

Figure 4.10 – Faute de saut sur le GPS : Distance entre les positions des deux filtres de Kalman

Nous voyons dans la figure 4.11 que le système de navigation inertiel (INS) et les encodeurs des roues arrières R-WSS ont des sorties similaires, tandis que la différence entre la position fournie par le GPS et celle délivrée par le système

odométrique avant F-odo dépasse le seuil $Thrs_{Pos_{GPS,F-odo}}$. Cela indique que l'erreur détectée dans le système est une erreur matérielle liée soit au GPS soit aux encodeurs avant du véhicule.

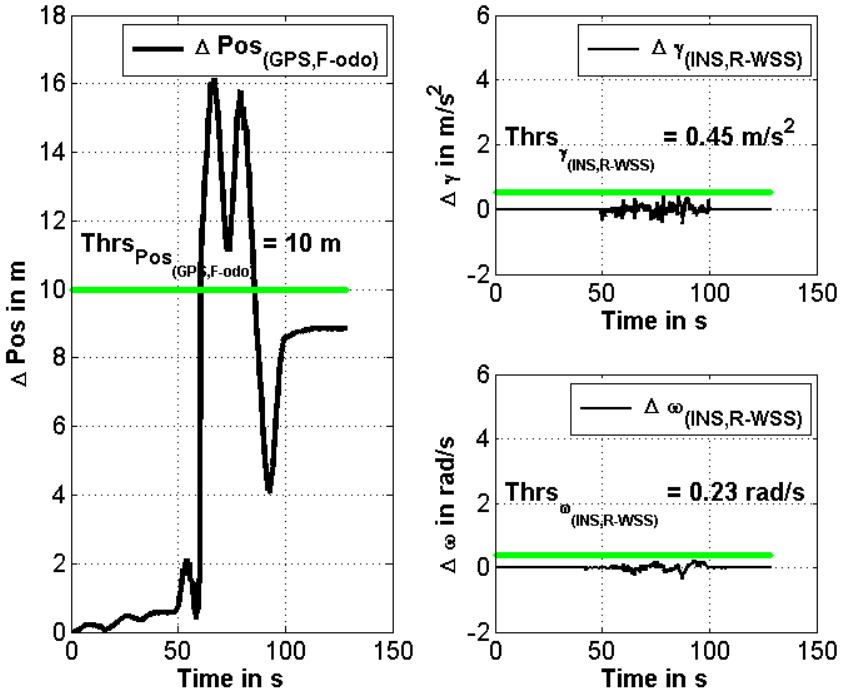

Figure 4.11 – Faute de saut sur le GPS : Comparaison des sorties des capteurs

La Figure 4.12 confirme que la valeur absolue du résidu issu de KF_2 ($|\ Res_{KF_2}\ |$) est nettement plus élevée que la valeur absolue du résidu issu de KF_1 ($|\ Res_{KF_1}\ |$) indiquant que la branche KF_2 contient une erreur.

Connaissant maintenant le couple (GPS, F-odo) contenant le capteur défectueux et la branche erronée, le système peut identifier que le capteur erroné est le GPS, et prendre la sortie correcte Pos_{KF_1} à t = 62,5 secondes, comme indiqué dans la figure 4.13, soit 2 secondes après l'injection de la faute.

A noter que le résidu Res_{KF_1} a une valeur importante seulement pendant un délai de 3 seconde. Après cela, il reprend une valeur inférieure à notre seuil de diagnostic car les écarts ont été lissés par le filtre. Il est donc vital de détecter l'erreur dans cette fenêtre, et la détermination des seuils, dépendant de l'application, est donc très importante.

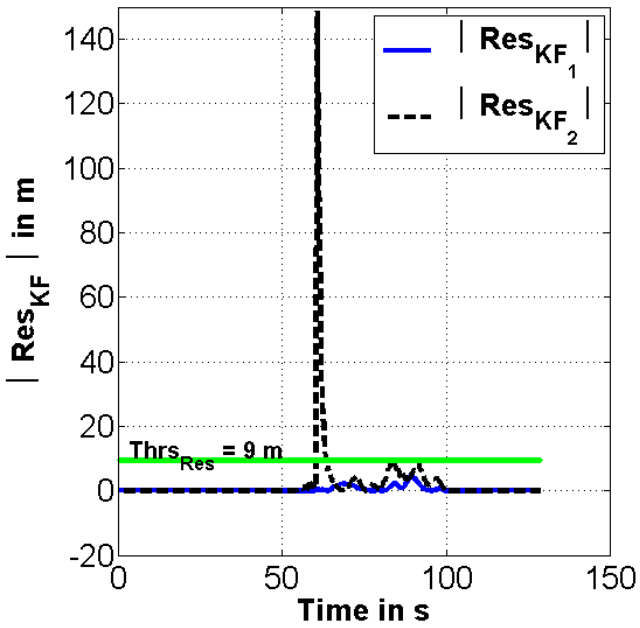

Figure 4.12 – Faute de saut sur le GPS : Comparaison des résidus

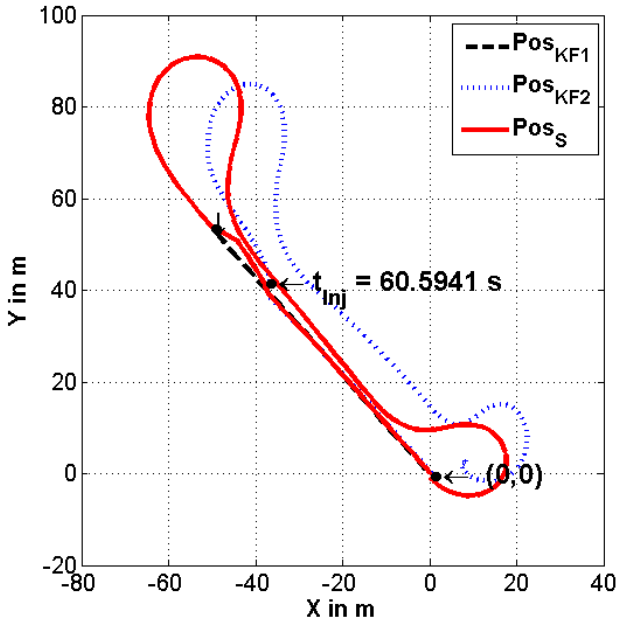

Figure 4.13 – Faute de saut sur le GPS : La position estimée par les deux filtres de Kalman et le système

- **Faute de mise à zéro de l'angle de braquage**

  La seconde faute injectée est une faute sur l'angle de braquage. Cette mesure est utilisée dans le système d'odomètrie avant *F-odo*. C'est une faute typique, mais grave dans de tels capteurs. Nous injectons la faute à l'instant $t_{Inj} = 59,7$

secondes, alors que le véhicule est conduit encore en ligne droite.

Les Figures 4.14, 4.15, 4.16 et 4.17 montrent que la faute est correctement détectée, diagnostiquée et le système est bien rétabli à t = 67.3 secondes, soit de 7.6 secondes après l'injection de la faute. En effet, le couple (GPS, F-odo) une fois de plus dépasse son seuil, mais cette fois | Res_{KF_1} | est supérieur à | Res_{KF_2} | indiquant que le F-odo est erroné. Le laps de temps pour la détection est dû au fait que, bien que la faute soit injectée à t_{Inj} = 59.7 secondes, elle n'a pas d'incidence sur le comportement du système avant que le véhicule commence à tourner (environ t = 65 secondes).

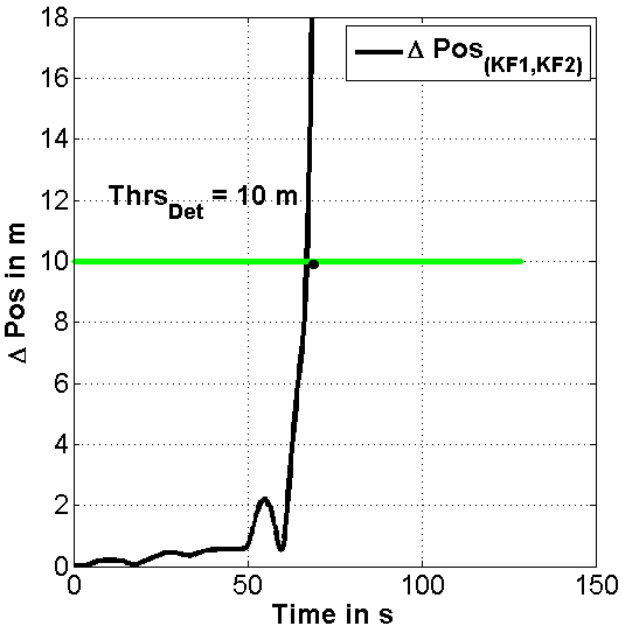

Figure 4.14 – Faute de mise à zéro sur l'angle de braquage : Distance entre les positions des deux filtres de Kalman

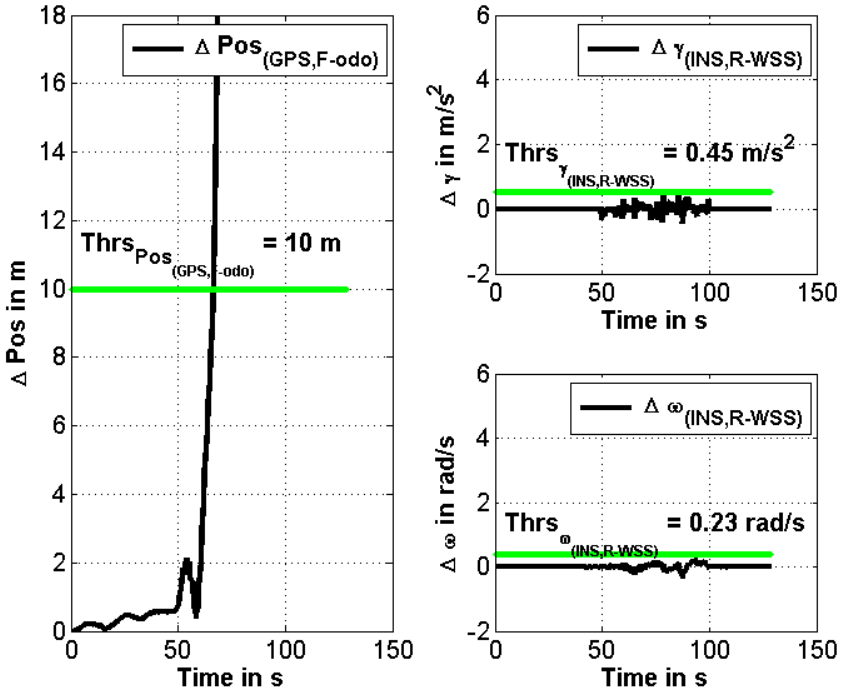

Figure 4.15 – Faute de mise à zéro sur l'angle de braquage : Comparaison des sorties des capteurs

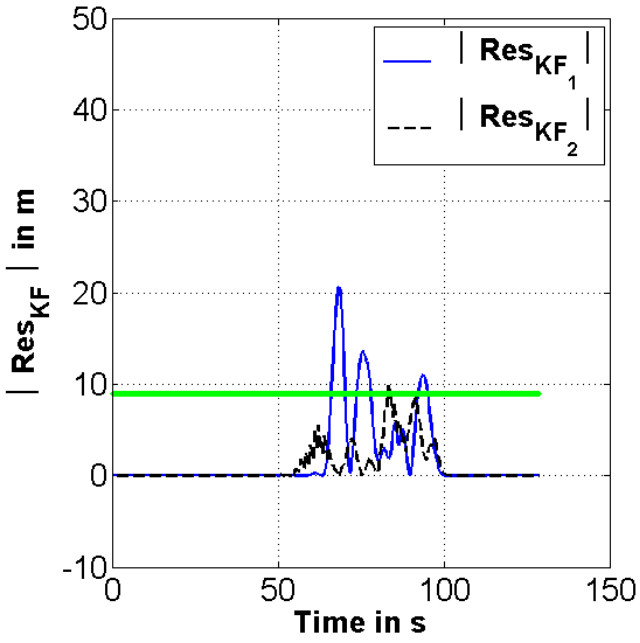

Figure 4.16 – Faute de mise à zéro sur l'angle de braquage : Comparaison des résidus

- **Faute de biais sur les encodeurs F-WSS**
  La troisième faute injectée est une faute de biais sur les encodeurs, en ajoutant un biais de 1 mètre/seconde sur la sortie du capteur odométrique F-WSS.

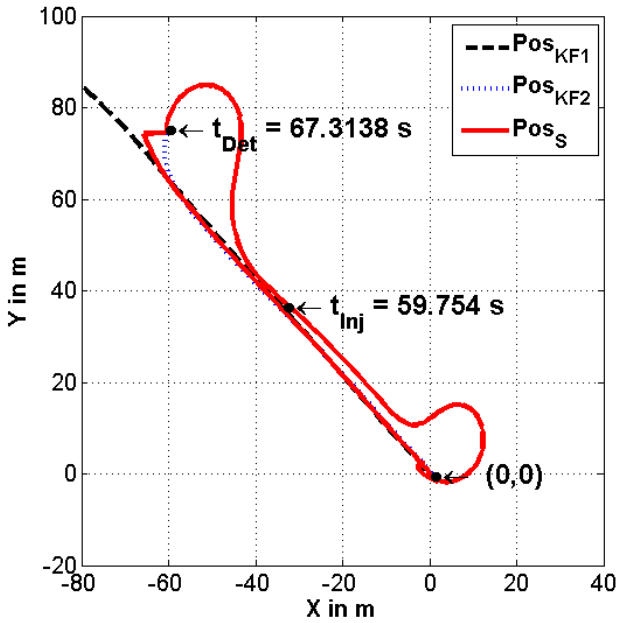

Figure 4.17 – Faute de mise à zéro sur l'angle de braquage : La position estimée par les deux filtres de Kalman et le système

La figure 4.18 montre que la faute injectée est correctement détectée. La figure 4.19 indique que la faute détectée est une faute matérielle liée au capteur GPS ou F-WSS. Cependant les deux résidus $\mid Res_1 \mid$ et $\mid Res_2 \mid$ (Figure 4.20) ne dépassent pas le seuil $Thrs_{Res}$ prédéfini. Dans ce cas l'erreur évolue trop lentement pour avoir une évolution rapide des résidus et par conséquent la comparaison des résidus ne fait pas ressortir l'erreur. Cela met en évidence une nouvelle limite de l'utilisation des résidus pour la détection de fautes car cela peut conduire comme ici à un diagnostic incomplet. Ainsi nous n'arrivons pas à distinguer lequel des capteurs (GPS ou F-WSS) est fautif, et par conséquent la branche erronée. Pour mettre le système en état sûr, une alerte est générée indiquant que le système de localisation mis en place est défaillant comme montré sur la figure 4.21.

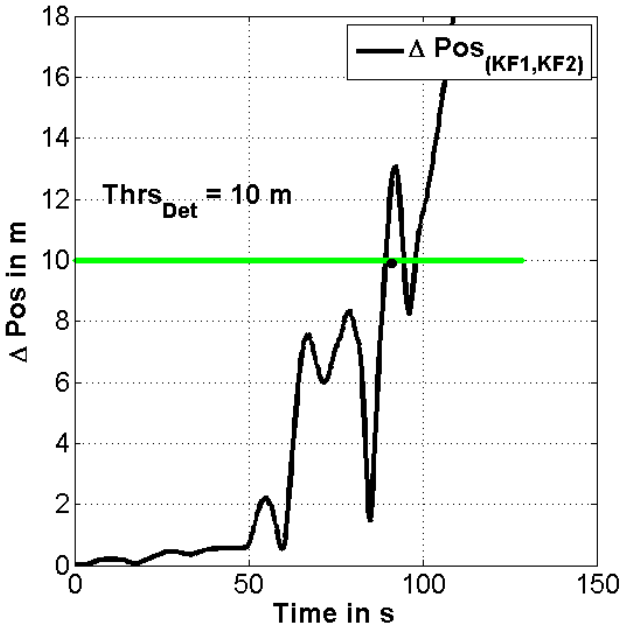

Figure 4.18 – Faute de biais sur les encodeurs F-WSS : Distance entre les positions des deux filtres de Kalman

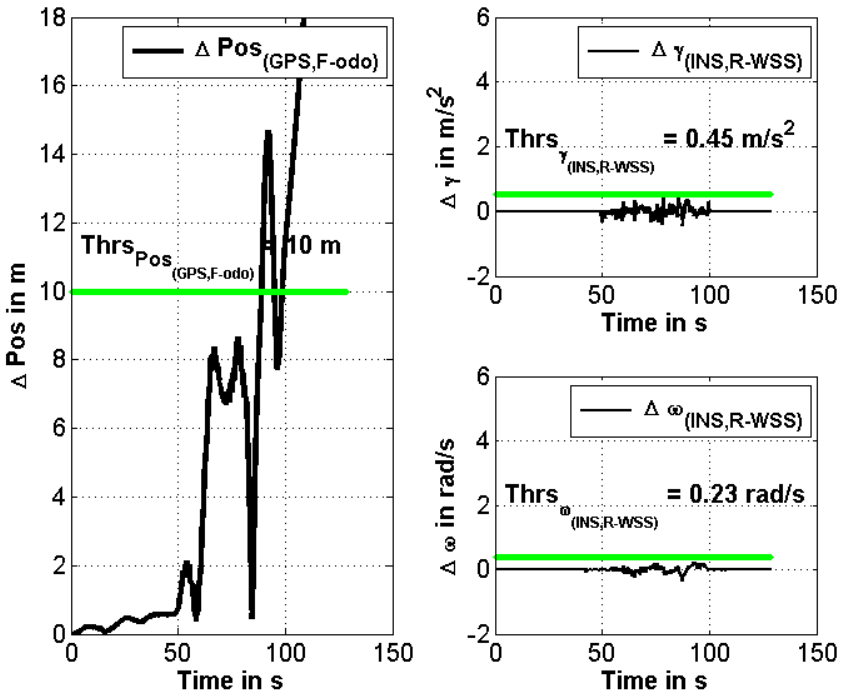

Figure 4.19 – Faute de biais sur les encodeurs F-WSS : Comparaison des sorties des capteurs

- **Faute logicielle dans le filtre de Kalman** KF_1

  La quatrième faute injectée est une faute logicielle dans le premier filtre de

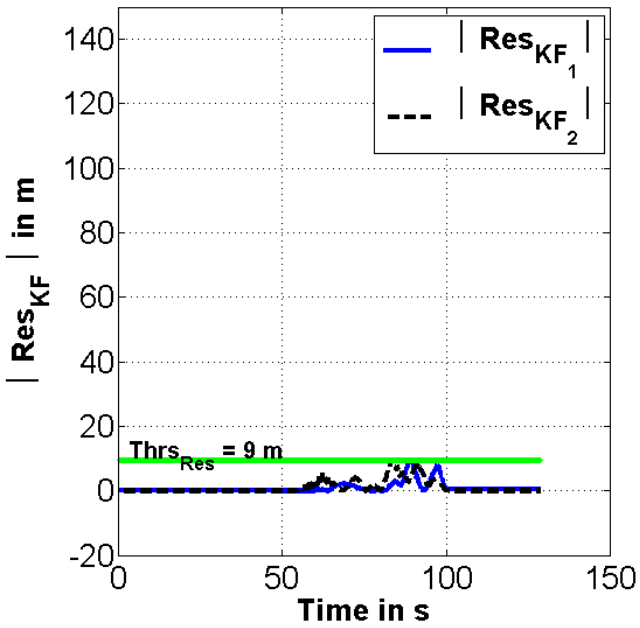

Figure 4.20 – Faute de biais sur les encodeurs F-WSS : Comparaison des résidus

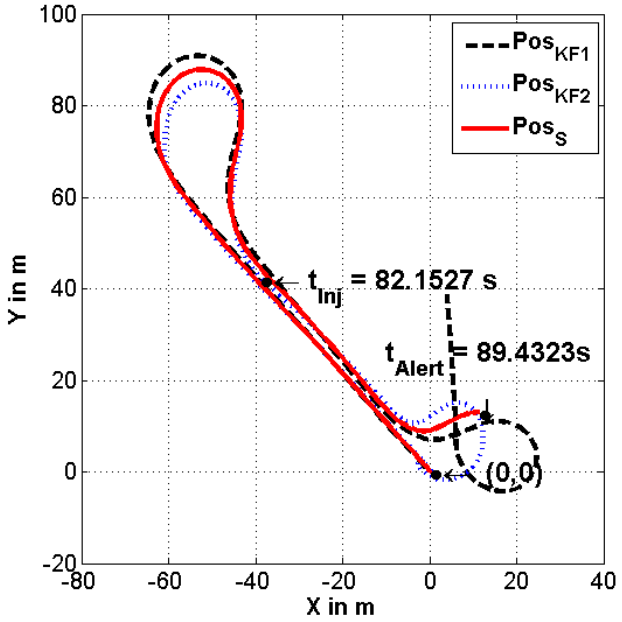

Figure 4.21 – Faute de biais sur les encodeurs F-WSS : La position estimée par les deux filtres de Kalman et le système

Kalman KF_1. Comme mentionné précédemment, les mécanismes de filtrage de Kalman nous paraissent particulièrement sensibles aux fautes de conception. Les covariances de bruits de modèle et de mesure Q et R ont un grand impact sur l'estimation de la sortie du filtre, et elles sont généralement déterminées empiriquement. Dans ce scénario, nous mettons Q = R/10000 dés le début de

l'expérience ($t_{Inj} = 0$) pour donner plus de confiance au système INS que le système d'odométrie avant F-odo.

Comme on le voit sur la figure 4.22, la position estimée par KF_1 dérive lentement de celle estimée par KF_2 avant de diverger brusquement et dépasser notre seuil de détection a l'instant t = 87,1 secondes.

La Figure 4.23 montre que la défaillance détectée n'est pas diagnostiquée comme une faute matérielle dans les capteurs de perception, indiquant ainsi une faute de logiciel dans l'un des algorithmes de fusion de données par filtrage de Kalman (KF_1 ou KF_2) sans être en mesure de distinguer lequel, par manque de redondance logicielle. Une application permettant de tolérer une telle faute est présentée au chapitre 5.

Après la détection de cette faute logicielle a l'instant $t_{Det} = 87.1$ secondes, notre architecture génère une alerte indiquant que le système de localisation est défaillant comme montré dans la figure 4.24. Il est particulièrement intéressant de voir qu'une telle faute ne peut pas être détectée pendant près de 80 secondes, car les résultats du bloc défaillant sont similaires à ceux du bloc non défaillant. Cela renforce notre opinion que les mécanismes de fusion de données sont difficiles à valider et que leurs fautes doivent être plus souvent prises en compte par des mécanismes de tolérance aux fautes.

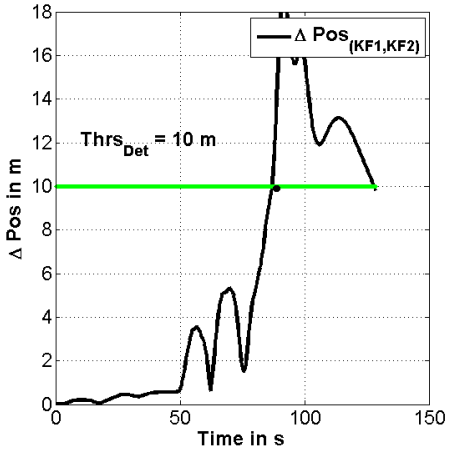

Figure 4.22 – Faute logicielle sur le filtre de Kalman 1 : Distance entre les positions des deux filtres de Kalman

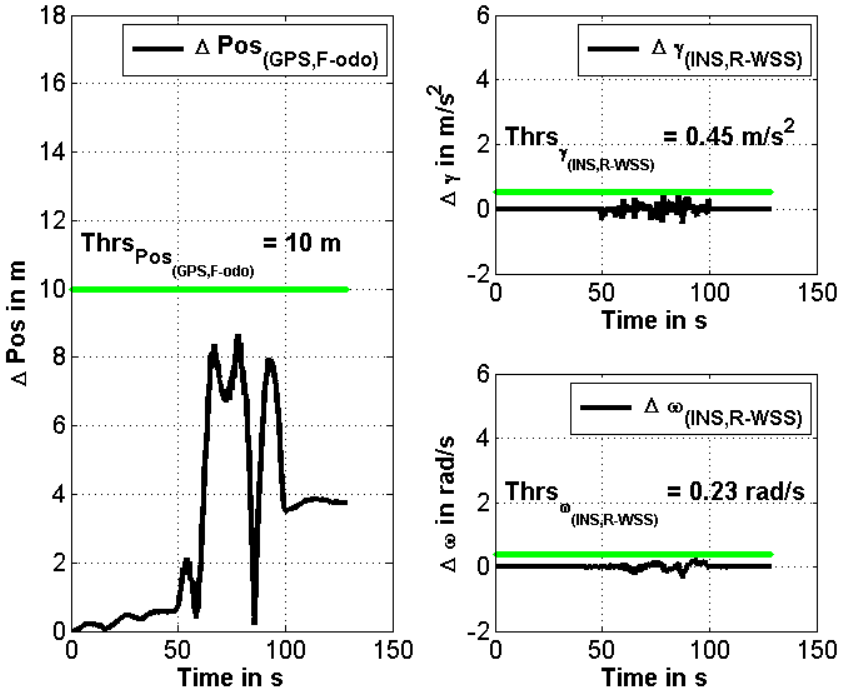

Figure 4.23 – Faute logicielle sur le filtre de Kalman 1 : Comparaison des sorties des capteurs

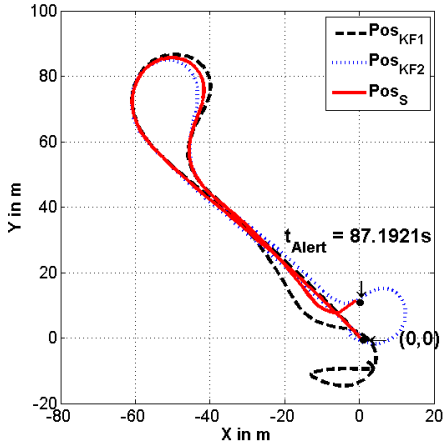

Figure 4.24 – Faute logicielle sur le filtre de Kalman 1 : La position estimée par les deux filtre de Kalman et le système

4.5 Résultats globaux sur la campagne d'expérimentation

Dans nos expériences, un grand nombre de fautes sur les capteurs et les algorithmes de filtre de Kalman ont été injectées. Pour chaque type de capteur, nous avons injecté un ensemble de fautes de manières différentes, à des instants différents, afin

de valider aussi largement que possible leurs conséquences sur le résultat final de l'architecture, et mesurer la capacité de notre approche en terme de détection et de rétablissement.

Cette section présente les résultats globaux de la campagne d'expérimentation, ainsi que les mesures exposées précédemment. Les résultats précis de chaque expérimentation se trouvent dans l'annexe A.

4.5.1 Mesures sur les fautes matérielles

- *Fautes sur le GPS* : Comme mentionné auparavant dans la section 3.2.1, le GPS souffre de plusieurs erreurs dues aux facteurs environnementaux qui interviennent dans la détermination d'une position d'un utilisateur donné. Ces facteurs peuvent causer des sauts dans la position fournie par le GPS. Pour modéliser les erreurs de positionnement par GPS et leurs effets, nous avons simulé quatre différents sauts (2, 5, 10, 20) injectés dans les quatre directions (+X, -X, +Y, -Y). Ces fautes sont injectées à quatre dates précises différentes. Ces différentes combinaisons font que le nombre de scénarios testés sur le GPS atteint soixante-quatre (64) scénarios. Un échantillon de résultats d'injection de fautes sur le GPS est montré dans le tableau 4.4 :

Comp	Saut	Dir	t_{Inj}	t_{Det}	Del_{Det}	Del_{Def}	$Del_{Def,Det}$	Del_{Err}	B_d	B_i	B_r	B_{Err}	B_{Def}
GPS	2	+X	60.59	X	X	X	X	X	0	0	0	0	0
GPS	5	-X	87.47	90.27	2.80	X	X	1.96	1	1	1	1	0
GPS	10	-Y	72.35	72.91	0.56	X	X	0.28	1	1	1	1	0
GPS	20	+Y	82.15	83.27	1.12	45.36	44.24	1.12	1	1	1	1	1

Tableau 4.4 – Injection de fautes sur le GPS

Notez que toutes les fautes injectées qui provoquent la défaillance du système sont correctement détectées, diagnostiquées, et tolérées.

- *Fautes sur les capteurs odométriques avant F-WSS* : pour simuler les fautes qui peuvent survenir sur les encodeurs, nous avons injecté les trois types de fautes modélisées dans la section 4.3.2.1. Les fautes de *mise à zéro*, et les fautes de *trame bloquée* sont injectées à quatre instants différents (ce qui fait 8 scénarios de fautes présentés dans les tableaux 4.5 et 4.6). Puis nous avons modélisé quatre biais de mesure (1, 2, 3, et 4) à quatre instants différents (ce qui fait 16 scénarios dont un échantillon est représenté dans le tableau 4.7). Ainsi vingt-quatre (24) scénarios de fautes sont simulés par l'injection de fautes sur le capteur F-WSS. Les trois tables 4.5, 4.6 et 4.7 donnent les résultats trouvés dans ces cas.

Comp	Type	t_{Inj}	t_{Det}	Del_{Det}	Del_{Def}	$Del_{Def,Det}$	Del_{Err}	B_d	B_i	B_r	B_{Err}	B_{Def}
F-WSS	Hard	60.59	64.51	3.92	13.44	9.52	0.56	1	1	1	1	1
F-WSS	Hard	72.35	77.67	5.32	11.48	6.16	-0.28	1	1	1	1	1
F-WSS	Hard	82.15	87.47	5.32	8.12	2.80	2.24	1	1	1	1	1
F-WSS	Hard	87.47	99.79	12.32	13.72	1.40	5.04	1	1	1	1	1

Tableau 4.5 – Injection de faute de type *trame bloquée* sur les encodeurs F-WSS

On remarque bien que toutes les fautes de type *trame bloquée* sont détectées, diagnostiquées, et que le système est bien rétabli.

Comp	Type	t_{Inj}	t_{Det}	Del_{Det}	Del_{Def}	$Del_{Def,Det}$	Del_{Err}	B_d	B_i	B_r	B_{Err}	B_{Def}
F-WSS	Null	60.59	62.27	1.68	2.52	0.84	1.12	1	1	1	1	1
F-WSS	Null	72.35	73.47	1.12	4.20	3.08	0.28	1	1	1	1	1
F-WSS	Null	82.15	82.71	0.56	2.52	1.96	0.00	1	1	1	1	1
F-WSS	Null	87.47	90.83	3.36	X	X	2.52	1	1	0	1	0

Tableau 4.6 – Injection de faute de type *mise à zéro* sur les encodeurs F-WSS

On remarque que les fautes de types *mise à zéro* sont détectées, et le type de capteur erroné est bien diagnostiqué. Cependant pour la dernière faute injectée, le rétablissement n'a pas eu lieu, étant donné que les deux résidus générés par les deux filtres KF_1 et KF_2 ne dépassent pas le seuil choisi $Thrs_{Res}$. Le diagnostic n'est pas complet et le capteur erroné n'est pas identifié, ainsi le système n'a pas pu être rétabli. Pour mettre le système en état sûr, notre architecture génère une alerte indiquant que le système de localisation est défaillant.

Comp	Bias	t_{Inj}	t_{Det}	Del_{Det}	Del_{Def}	$Del_{Def,Det}$	Del_{Err}	B_d	B_i	B_r	B_{Err}	B_{Def}
F-WSS	1	72.35	85.23	12.88	33.88	21.00	10.08	1	1	0	1	1
F-WSS	2	60.59	63.39	2.80	14.56	11.76	1.40	1	1	1	1	1
F-WSS	3	82.15	86.07	3.92	15.96	12.04	2.80	1	1	1	1	1
F-WSS	4	87.47	89.15	1.68	8.12	6.44	0.84	1	1	1	1	1

Tableau 4.7 – Injection de fautes de type *biais* sur les encodeurs F-WSS

On remarque que toutes les fautes de *biais* injectée sur les encodeurs du robot sont détectées, et diagnostiquées. Cependant pour un biais de 1 mètre le rétablissement n'est pas effectué, car ce biais ne permet pas au résidu généré par les filtres de Kalman d'être significatif, par contre pour des biais supérieurs à cette valeur le rétablissement a bien eu lieu.

- *Faute sur l'angle de braquage* : nous avons seulement simulé deux fautes de *mise à zéro* sur l'angle de braquage. Nous avons injecté ces fautes pendant que le véhicule traverse la première et la seconde ligne droite du circuit d'expérimentation. Pendant cette période la faute n'est pas détectée car elle n'a pas de conséquence significative sur le système. Elle est détectée au moment où le véhicule tourne pour faire le premier et le second rond point respectivement. La table 4.8 résume les paramètres d'évaluation de ces fautes.

Comp	Type	t_{Inj}	t_{Det}	Del_{Det}	Del_{Def}	$Del_{Def,Det}$	Del_{Err}	B_d	B_i	B_r	B_{Err}	B_{Def}
Angle de braquage	Null	59.75	67.31	7.56	9.52	1.96	1.40	1	1	1	1	1
Angle de braquage	Null	82.15	88.03	5.88	13.16	7.28	1.12	1	1	1	1	1

Tableau 4.8 – Injection de fautes sur le capteur d'angle de braquage

D'après les traces d'exécution et les différents relevés, sur les quatre-vingt-dix (90) fautes matérielles injectées sur les capteurs GPS, F-WSS, et l'angle de braquage, toutes les fautes détectées sont diagnostiquées, cependant seulement 62.32 % sont recouvertes et permettent au système d'être rétabli. Cet écart de performance est dû principalement au choix des seuils de rétablissement utilisés. En effet, nous avons choisi empiriquement tous les seuils de façon à ce qu'ils excédent les valeurs de comportement nominal. Pour améliorer les performances de notre architecture une étude approfondie de ces seuils devient indispensable. Nous remarquons aussi qu'aucun *faux positif* n'est engendré par notre architecture sur cet ensemble de fautes matérielles, et aucune *absence de détection* n'est signalée ainsi que déterminé théoriquement. 44% de fautes détectées entrainent une erreur dans le système ($Diff_{CF,CN} \geq Thrs_{Err}$) sans que le système soit défaillant ($Diff_{CF,CN} \geq Thrs_{Def}$). Pour ces fautes la détection est correcte il y a bien une erreur mais ces erreurs n'ont pas causé de défaillance et pourraient à la rigueur être considérées comme des détections intempestives. Ainsi que dit à la section 4.3.4, ce sera au concepteur de décider quel comportement du système définir face à ces fautes. A noter que des perturbations externes importantes sur les capteurs (par exemple reflet de soleil sur une camera, des saut GPS, bruit de mesure sur les centrales inertielles) peuvent se produire et pourraient être traitées soit en relevant également le seuil de détection, soit en les considérant comme des fautes temporaires qui disparaitraient du système en même temps que la perturbation.

La table 4.9 donne un aperçu des mesures globales sur toutes les différentes fautes matérielles injectées dans notre campagne d'expérimentation.

Type de faute	P_i (%)	P_r (%)	P_{FP} (%)	P_{ND} (%)	P_{ESD}(%)
Matérielle	100.00	62.32	0.00	0.00	44

Tableau 4.9 – Mesures sur les fautes matérielles

4.5.2 Mesures sur les fautes logicielles

Pour les fautes logicielles, de nombreuses variations de mutations peuvent être réalisées. Pour montrer l'effet de ces mutations sur notre architecture, nous avons testé un échantillon de vingt (20) différentes mutations qui couvre tous les types

de mutations possibles présentées en section 4.3.2.2. Ces mutations modélisent des erreurs dans le modèle du système et/ou d'observation utilisés dans le filtre de Kalman.

Les 20 mutations que nous avons testées représentent notre ensemble de fautes logicielles injectées, et se répartissent comme suit :

- 8 substitutions de valeurs numériques,
- 4 substitutions d'expressions géométriques,
- 2 substitutions de signe d'une variable,
- 4 substitutions de variable,
- 2 mutations modélisant l'erreur sur les matrices de covariances Q et R.

Ces différentes mutations ont été générées manuellement, en modifiant le code exécuté par les deux filtres de Kalman. Le tableau 4.10 montre les effets de ces erreurs logicielles, et la capacité de notre approche en termes de détection de ce type d'erreurs.

Original	Mutant	t_{Inj}	t_{Det}	Del_{Det}	Del_{Def}	$Del_{Def,Det}$	Del_{Err}	B_d	B_i	B_r	B_{Err}	B_{Def}
Q1	Q1 = 0.00001.R1	0.00	87.19	87.19	X	X	25.76	1	0	0	1	0
$A^1_{1,4} : cos(\theta)$	$A^1_{1,4} : sin(\theta)$	0.00	72.63	72.63	X	X	19.32	1	0	0	1	0
$A^1_{2,4} : sin(\theta)$	$A^1_{2,4} : cos(\theta)$	0.00	71.23	71.23	X	X	18.20	1	0	0	1	0
$A^1_{1,4} : cos(\theta)$	$A^1_{1,4} : -cos(\theta)$	0.00	72.07	72.07	X	X	18.76	1	0	0	1	0
$A^1_{2,4} : sin(\theta)$	$A^1_{2,4} : -sin(\theta)$	0.00	72.07	72.07	X	X	19.04	1	0	0	1	0
$B^1_{1,1} : \frac{1}{2}dt^2$	$B^1_{1,1} : dt^2$	0.00	X	X	X	X	X	0	0	0	0	0
$B^1_{1,1} : \frac{1}{2}dt^2$	$B^1_{1,1} : \frac{1}{4}dt^2$	0.00	X	X	X	X	X	0	0	0	0	0
$H^1_{1,1} : 1$	$H^1_{1,1} : 0$	0.00	88.03	88.03	89.99	1.96	21.28	1	0	0	1	1
$H^1_{1,2} : 0$	$H^1_{1,2} : 1$	0.00	56.11	56.11	58.07	1.96	2.52	1	0	0	1	1
$H^1_{1,1} : 1$	$H^1_{1,1} : -1$	0.00	53.87	53.87	55.83	1.96	1.40	1	0	0	1	1
Q2	Q2 = 0.00001R2	0.00	84.67	84.67	X	X	24.92	1	0	0	1	0
$A^2_{1,1} : 1$	$A^2_{1,1} : 0$	0.00	55.83	55.83	57.79	1.96	3.08	1	0	0	1	1
$A^2_{1,1} : 1$	$A^2_{1,1} : -1$	0.00	55.83	55.83	57.79	1.96	3.08	1	0	0	1	1
$A^2_{1,2} : 0$	$A^2_{1,2} : 1$	0.00	55.83	55.83	57.79	1.96	3.36	1	0	0	1	1
$B^2_{1,1} : cos\theta$	$B^2_{1,1} : sin\theta$	0.00	57.79	57.79	X	X	5.60	1	0	0	1	0
$B^2_{2,1} : sin\theta$	$B^2_{2,1} : cos\theta$	0.00	57.79	57.79	X	X	5.32	1	0	0	1	0
$B^2_{1,1} : dt$	$B^2_{1,1} : dt/2$	0.00	X	X	X	X	5.32	0	0	0	1	0
$H^2_{1,1} : 1$	$H^2_{1,1} : 0$	0.00	X	X	X	X	5.32	0	0	0	1	0
$H^2_{1,2} : 0$	$H^2_{1,2} : 1$	0.00	55.55	55.55	58.35	2.80	2.52	1	0	0	1	1
$H^2_{1,1} : 1$	$H^2_{1,1} : -1$	0.00	54.71	54.71	55.83	1.12	2.24	1	0	0	1	1

Tableau 4.10 – Injection de fautes logicielles par mutation

D'après les traces d'exécution, sur les 20 mutations que nous avons simulées 90 % des fautes ayant une incidence sur le système ($B_{Err} = 1$) ont été détectées par notre architecture. Comme mentionné à plusieurs reprises, étant donné que le niveau de redondance dans notre architecture est limité à deux branches, ces fautes logicielles détectées ne sont pas rétablies. Pour assurer ce service de rétablissement une troisième branche diversifiée est nécessaire, tel que présenté dans le chapitre 5. Pour ces fautes logicielles aucun *faux positif* n'est engendré et aucune *non*

détection n'est signalée. 44% de mutations réalisées engendrent une erreur dans le système sans sa défaillance. Ces erreurs comme mentionné précédemment dans les fautes matérielles pourraient être considérées comme des détections intempestives qui pourront être traitées selon deux solutions :

- prendre en compte les perturbations externes importantes qui peuvent survenir sur les capteurs (par exemple reflet de soleil sur une caméra, des saut GPS, bruit de mesure sur les centrales inertielles) en relevant le seuil de détection.

- considérer des fautes temporaires qui disparaitraient du système en même temps que les perturbations.

La table 4.11 résume les mesures globales sur les différentes mutations simulant des fautes logicielles réalisées.

Type de faute	P_i (%)	P_r (%)	P_{FP} (%)	P_{ND} (%)	P_{ESD}(%)
Logicielles			0.00	0.00	44

Tableau 4.11 – Mesures sur les fautes logicielles

4.6 Discussion globale

D'après notre expérimentation et les résultats obtenus nous dégageons les remarques suivantes :

- Notons que la détection d'erreur n'est pas instantanée après l'injection de la faute, étant donné que la sortie du bloc du filtre de Kalman erroné prend un certain temps pour s'écarter de la sortie correcte. Certaines fautes sont en effet latentes, comme on le voit dans la seconde injection de notre exemple d'expérimentation présenté dans la section 4.4.3, lorsque l'erreur sur l'angle de braquage n'a pas été activée jusqu'à ce que le véhicule tourne. Cela ne soulève aucun problème en ce qui concerne la sortie du système car pendant ce temps la différence entre les deux sorties correcte et erronée des blocs de filtrage de Kalman n'est évidemment pas supérieure à $Thrs_{Det}$, une valeur admissible. Toutefois, sans l'hypothèse de faute unique que nous avons considéré dans ce travail, l'activation d'une autre faute pendant cette période pourrait rendre le système incapable de détecter et diagnostiquer la première erreur (et par conséquent la seconde).

- Notons aussi que la comparaison des résidus serait suffisante pour détecter le bloc erroné pour les fautes matérielles comme une alternative à la distance entre les positions que nous avons choisi d'utiliser (figures 4.12 et 4.16). Cependant, nous estimons qu'il est très dépendant des covariances des bruits Q et R dans le modèle de filtre de Kalman, et une erreur non détectée dans les mécanismes de fusion de données peut entraîner des *faux positifs* ou de *non détection*. De plus l'utilisation des résidus pour la détection de fautes matérielles sur les capteurs n'est pas une bonne idée pour nous, car cela revient à utiliser les mécanismes de fusion de données pour la détection de fautes alors que nous ne leur faisons pas confiance. En effet les mécanismes de fusion de données sont difficiles à développer et à valider. Dans nos expériences nous avons identifié deux cas où la comparaison de résidus pose problème : la première (figure 4.12) montre que le conflit peut disparaitre rapidement et donc risquer de ne pas etre détecté si la fréquence d'échantillonnage est trop faible. Dans le second cas (figure 4.20) l'erreur évolue trop lentement par rapport au comportement nominal pour que la comparaison de résidus puisse le détecter significativement. Cependant si les algorithmes de fusion de données sont validés formellement, une utilisation des résidus pour la détection de faute capteurs devient possible et faisable. Mais ainsi que mentionné dans la section 2.5 cette validation formelle peut être impossible.

- D'après notre expérimentation, nous constatons qu'une limite principale de nos mécanismes proposés est la détermination des différents seuils utilisés pour la détection, le diagnostic et le recouvrement. Dans ce travail nous ne nous sommes pas concentrés sur cette contrainte classique aux mécanismes de détection, et nous avons déterminé ces seuils empiriquement de manière à ce qu'ils excédent les valeurs de comportement nominal du système. Cependant déterminer les valeurs de ces seuils est un problème intéressant a traiter, et il sera l'une de nos préoccupations dans nos futurs travaux.

4.7 Conclusion

Dans ce chapitre nous avons présenté une évaluation des performances des mécanismes de tolérance aux fautes présentés dans le chapitre 3. Cette validation est basée sur l'acquisition de données réelles, la méthode de rejeu de données et la technique d'injection de fautes sur ces données.

Nous avons présenté la campagne d'expérimentation réalisée, cette campagne est un processus long et couteûx, particulièrement dans des systèmes de perception.

Les expériences réalisées montrent de façon objective que, dans l'ensemble des fautes matérielles et logicielles injectées que nous avons considérées, les mécanismes de tolérance aux fautes implémentés donnent d'excellents résultats en terme de détection d'erreur. Le rétablissement du système est bon (62.32 %) pour les fautes matérielles. Le rétablissement pour les fautes logicielles est présenté dans le chapitre suivant.

Le nombre d'expériences que nous avons réalisées 110 ne couvre pas toutes les fautes que notre système peut rencontrer, cependant nous estimons que la portion de fautes injectées représente bien la plupart des fautes réelles auxquelles peut être confronté notre système de localisation. De plus, notre contexte d'exécution ne représente qu'un exemple d'application de fusion de données dans le domaine robotique, il serait intéressant d'adapter nos mécanismes de tolérance aux fautes pour d'autres types d'application telles que la détection d'obstacles, le suivi d'objets ...

Tolérance aux fautes logicielles par diversification de filtres de Kalman

Sommaire

5.1 Introduction

Nous présentons ici des mécanismes de tolérance aux fautes logicielles permettant le rétablissement du système pour l'estimation d'orientation des robots mobiles.

La *diversification fonctionnelle* est une méthode issue de la sûreté de fonctionnement qui a pour objectif de détecter et tolérer des fautes. Dans ce chapitre, nous présentons en détail une diversification fonctionnelle des filtres de Kalman, par le biais de la programmation N-versions. Nous implémentons trois filtres de Kalman diversifiés, et nous utilisons la méthode de vote majoritaire pour tolérer des fautes logicielles en fusion de données.

Nous rappelons d'abord les principes de diversification fonctionnelle, nous présentons par la suite les trois filtres de Kalman, et nous détaillons les services de détection et de rétablissement offerts. Finalement nous évaluons les performances de ces mécanismes de la même manière que dans le chapitre 4.

5.2 Principes de la diversification fonctionnelle

La diversification fonctionnelle est l'un des moyens de tolérance aux fautes permettant d'assurer la continuité du service d'un système. Cette technique est largement utilisée pour la détection et la tolérance aux fautes de conception et de développement. Elle repose sur l'utilisation de plusieurs exemplaires d'un composant, appelés variantes, qui mettent en œuvre la même fonction via des conceptions et des réalisations séparées à partir de la même spécification. Un décideur est implémenté à partir des sorties des différentes variantes pour fournir un résultat supposé exempt d'erreur. Il existe trois techniques de base de tolérance aux fautes par diversification fonctionnelle : les blocs de recouvrement [Randell, 1975], la programmation en N-versions [Avizienis and Chen, 1977] et la programmation N-autotestable [Laprie et al., 1990].

Dans *les blocs de recouvrement*, les variantes sont appelées des alternants et le décideur est un test d'acceptation, qui est appliqué séquentiellement aux résultats fournis par les alternants. Si les résultats fournis par le premier alternant ne sont pas satisfaisants, le deuxième alternant est exécuté, et ainsi de suite jusqu'à la satisfaction du test d'acceptation ou l'épuisement des alternants disponibles, auquel cas le bloc de recouvrement est globalement défaillant.

Dans *la programmation N-autotestable*, au moins deux composants autotestables sont exécutés en parallèle, chacun étant constitué soit de l'association d'une variante et d'un test d'acceptation, soit de deux variantes et d'un algorithme de comparaison.

Dans *la programmation N-versions*, les variantes sont appelées des versions et le décideur effectue un vote majoritaire sur les résultats de toutes les versions.

C'est cette dernière techniques que nous employons dans la suite de ce chapitre pour tolérer des fautes logicielles en fusion de données par filtrage de Kalman.

Notons que les deux filtres de Kalman du chapitre 3 étaient déjà un exemple de programmation 2-versions utilisé pour la détection de fautes logicielles. Dans ce chapitre, nous proposons une architecture 3-versions afin de non-seulement détecter ces fautes, mais aussi les tolérer.

5.3 Mécanismes proposés

Les mécanismes que nous proposons se basent principalement sur la diversification des modèles de processus et des entrées de filtre de Kalman. Cette diversification permet, en théorie, de tolérer des fautes logicielles au niveau du modèle du filtre de Kalman essentiel dans l'estimation des paramètres de sortie du système. Leur principe général consiste à exécuter plusieurs filtres parallèles, disposant du modèle

et d'entrées diversifiés.

Dans ce chapitre, nous implémentons une application d'estimation d'un angle de lacet θ d'un robot mobile. Trois versions parallèles sont mis en œuvres (figure 5.1), chacune prenant des entrées différentes, et exécutant un filtre de Kalman diversifié. Le premier filtre combine les sorties du capteur odométrique arrière R-WSS, et d'un gyroscope $Gyro_1$. Le second filtre combine les sorties d'un capteur odométrique avant F-WSS et d'un second gyroscope $Gyro_2$. Le troisième filtre combine la sortie de l'un des gyroscopes ($Gyro_1$ ou $Gyro_2$) et les sorties odométriques arrière. Cette diversification fonctionnelle permet à la fois de tolérer une faute matérielle dans l'un des capteurs, mais aussi une faute logicielle dans l'un des filtres de Kalman. Nous nous concentrons dans ce chapitre sur l'évaluation de la tolérance aux fautes logicielles. Les détails d'implémentation de ces trois filtres sont exposés ci-dessous au 5.4, et les services de détection et de rétablissement sont présentés dans la section 5.5.

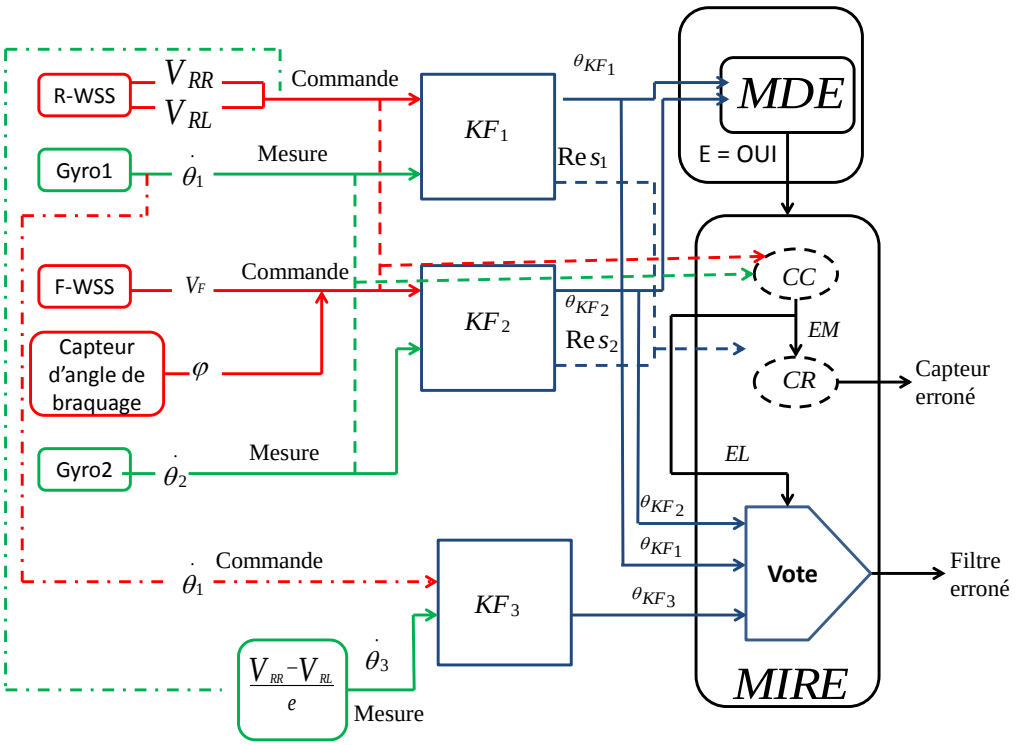

Figure 5.1 – Diversification fonctionnelle pour la tolérance aux fautes logicielles

Dans la figure 5.1 :

- MDE : Module de Détection d'Erreur
- MIRE : Module d'Identification et de Recouvrement d'Erreur

- CC : Comparaison des Capteurs
- CR : Comparaison des Résidus
- EM : Erreur Matérielle
- EL : Erreur Logicielle

5.4 Trois filtres de Kalman diversifiés

L'orientation des robots mobiles est un paramètre important dans les applications de localisation et de navigation. Les capteurs gyroscopiques et odométriques sont généralement les plus utilisés pour l'estimation de ce paramètre. Cependant le gyroscope et l'odomètre produisent une dérive lorsqu'ils sont utilisés seuls. Toutefois, ils peuvent êtres combinés par fusion de données et donc diminuer la dérive du système. Par exemple le gyroscope peut compenser l'odométrie lorsque les roues glissent dans les virages difficiles ou lors de roulement sur des bosses du terrain.

Dans cette section, pour estimer l'orientation du robot θ, nous utilisons tous les capteurs de mesure (capteur de vitesse arrière *R-WSS*, ceux de devant *F-WSS*, le gyroscope, et le capteur d'angle de braquage). Nous combinons les sorties de tous ces capteurs par les mécanismes de filtrage de Kalman pour estimer avec précision l'angle de lacet du robot θ. Dans ce qui suit nous présentons trois filtres de Kalman diversifiés utilisant les sorties des capteurs cités précédemment pour estimer l'angle de lacet θ.

5.4.1 Filtre de Kalman 1 (KF_1) : Couplage $R-WSS$, $Gyro_1$

Une solution possible pour l'estimation de l'angle de lacet θ d'un robot mobile consiste à combiner les informations provenant du capteur de vitesse de roues arrières (R-WSS) et de gyroscope ($Gyro_1$) en utilisant le modèle cinématique de type char. La vitesse angulaire de lacet $\dot{\theta}_{KF_1}$ est d'abord déduite des capteurs de vitesse de roues arrières (V_{RR}, et V_{RL}), puis est utilisée pour estimer l'orientation du robot θ_{KF_1} lors de la phase de prédiction du filtre de Kalman. Cette estimation est ensuite corrigée par la sortie du $Gyro_1$ ($\dot{\theta}_1$) au cours de la phase de correction.

- **Modèle de processus** : Le vecteur d'état comprend l'orientation du robot mobile θ_{KF_1} et sa vitesse angulaire $\dot{\theta}_{KF_1}$. Ces paramètres sont estimés à partir de la sortie du capteurs odométriques arrières (V_{RR}, V_{RL}) selon les équations 5.1.

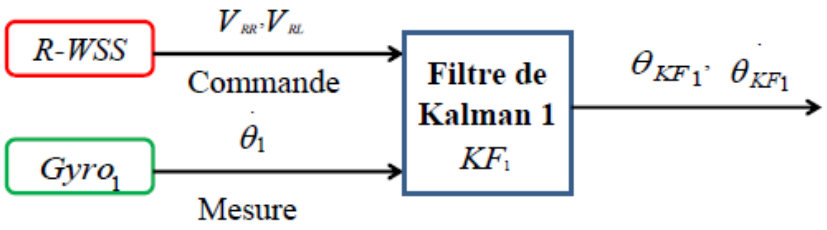

Figure 5.2 – Filtre de Kalman 1 (KF_1) : Couplage $R-WSS$, $Gyro_1$

$$\underbrace{\begin{bmatrix}\theta_{KF1_k}\\ \dot{\theta}_{KF1_k}\end{bmatrix}}_{X_{KF1_k}} = \underbrace{\begin{bmatrix}1 & 0\\ 0 & 0\end{bmatrix}}_{A_{KF1}} \cdot \underbrace{\begin{bmatrix}\theta_{KF1_{k-1}}\\ \dot{\theta}_{KF1_{k-1}}\end{bmatrix}}_{X_{KF1_{k-1}}} + \underbrace{\begin{bmatrix}\frac{\Delta t}{e} & \frac{-\Delta t}{e}\\ \frac{1}{e} & -\frac{1}{e}\end{bmatrix}}_{B_{KF1}} \cdot \underbrace{\begin{bmatrix}V_{RR_{k-1}}\\ V_{RL_{k-1}}\end{bmatrix}}_{U_{INS_{k-1}}} + w_{KF1_k} \tag{5.1}$$

où V_{RR} et V_{RL} sont respectivement la vitesse de la roue arrière droite et la vitesse de la roue arrière gauche, e est l'entraxe du véhicule (la distance entre les deux roues de l'essieu), et w_{KF1_k} est le bruit du modèle.

- **Modèle d'observation** : Le vecteur de mesure comprend la vitesse angulaire $\dot{\theta}_1$ émis par le $Gyro_1$. Cette mesure est liée au vecteur d'état du système par l'équation (5.2).

$$Z_{KF1_k} = \begin{bmatrix}\dot{\theta}_{1_k}\end{bmatrix} = \underbrace{\begin{bmatrix}0 & 1\end{bmatrix}_k}_{H_{KF1}} \cdot \underbrace{\begin{bmatrix}\theta_{KF1_k}\\ \dot{\theta}_{KF1_k}\end{bmatrix}}_{X_{KF1_k}} + v_{KF1_k} \tag{5.2}$$

où v_{KF1_k} est le bruit de mesure.

5.4.2 Filtre de Kalman 2 (KF_2) : Couplage $F-WSS$, $Gyro_2$

Le second algorithme du filtre de Kalman combine les informations du capteur odométrique avant (F-WSS) et l'angle de braquage de la roue avant ϕ pour prédire l'angle de lacet du robot θ. Cette estimation sera par la suite corrigée par la mesure du $Gyro_2$ dans l'étape de correction.

- **Modèle du processus** : Lors de la phase de prédiction du filtre de Kalman le modèle (5.3) est utilisé pour estimer l'orientation du robot mobile θ_{KF_2} et sa vitesse angulaire $\dot{\theta}_{KF_2}$ en combinant l'angle de braquage ϕ et la vitesse de la roue avant V_F.

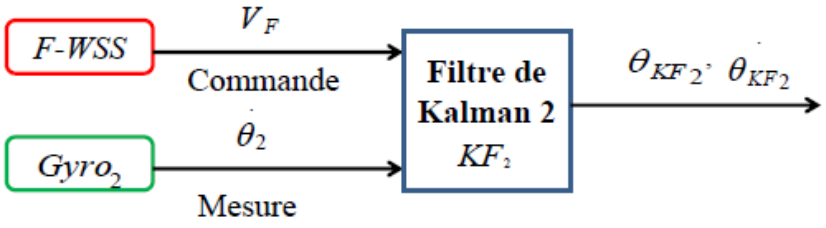

Figure 5.3 – Filtre de Kalman KF_2 : Couplage $F-WSS$, $Gyro_2$

$$\underbrace{\begin{bmatrix}\theta_{KF_{2_k}}\\ \dot{\theta}_{KF_{2_k}}\end{bmatrix}}_{X_{KF_{2_k}}} = \underbrace{\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}}_{A_{KF_2}} \cdot \underbrace{\begin{bmatrix}\theta_{KF_{2_{k-1}}}\\ \dot{\theta}_{KF_{2_{k-1}}}\end{bmatrix}}_{X_{KF_{2_{k-1}}}} + \underbrace{\begin{bmatrix}\frac{\Delta t sin\phi}{L}\\ \frac{sin\phi}{L}\end{bmatrix}}_{B_{KF_2}} \cdot \underbrace{\begin{bmatrix}V_{F_{k-1}}\end{bmatrix}}_{U_{FWSS_{k-1}}} + w_{KF_{2_k}} \tag{5.3}$$

où L est la distance entre l'axe avant et arrière du véhicule.

- **Modèle d'observation** : Lors de la phase de correction la mesure du $Gyro_2$ $\dot{\theta}_2$ est utilisée. Cette dernière est liée au vecteur d'état du système suivant l'équation (5.4).

$$Z_{KF_{1_k}} = \begin{bmatrix}\dot{\theta}_{2_k}\end{bmatrix} = \underbrace{\begin{bmatrix}0 & 1\end{bmatrix}_k}_{H_{KF_2}} \cdot \underbrace{\begin{bmatrix}\theta_{KF_{1_k}}\\ \dot{\theta}_{KF_{1_k}}\end{bmatrix}}_{X_{KF_{2_k}}} + v_{KF_{1_k}} \tag{5.4}$$

5.4.3 Filtre de Kalman 3 (KF_3) : Couplage de $Gyro_1$, R-WSS

Pour obtenir une troisième version du filtre de Kalman, nous forçons la diversification par rapport aux deux précédents filtres, ici (a) en prenant un couple de capteurs différents de ceux utilisés dans les deux autres filtres, et (b) en inversant la phase pour laquelle ils sont utilisés (correction, prédiction). Nous utilisons ainsi pour ce troisième filtre la sortie de l'un des gyroscopes en entrée de modèle lors de la phase de prédiction, et les sorties des odomètres arrière après un certain pré-traitement en phase de correction.

- **Modèle du processus** : Ce filtre de Kalman calcule dans l'étape de prédiction l'angle de lacet $\theta_{KF_{3_k}}$ en utilisant la vitesse angulaire issue de $Gyro_1$ $\dot{\theta}_1$ suivant l'équation 5.5.

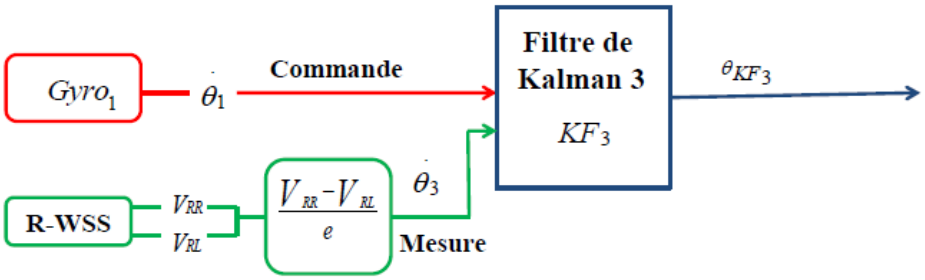

Figure 5.4 – Filtre de Kalman 3 (KF_3) : Couplage de $Gyro_1$,R-WSS

$$\underbrace{\begin{bmatrix}\theta_{KF3_k}\\ \dot{\theta}_{KF3_k}\end{bmatrix}}_{X_{KF3_k}} = \underbrace{\begin{bmatrix}1 & 0\\ 0 & 0\end{bmatrix}}_{A_{KF3}} . \underbrace{\begin{bmatrix}\theta_{KF3_{k-1}}\\ \dot{\theta}_{KF3_{k-1}}\end{bmatrix}}_{X_{KF3_{k-1}}} + \underbrace{\begin{bmatrix}\Delta t\\ 1\end{bmatrix}}_{B_{KF3}} . \underbrace{\begin{bmatrix}\dot{\theta}_1\end{bmatrix}}_{U_{RWSS_{k-1}}} + w_{KF3_k} \tag{5.5}$$

- **Modèle d'observation** : Pour corriger l'état du système $(\theta_{KF3_k}, \dot{\theta}_{KF3_k})$, nous utilisons cette fois ci les sorties odométriques arrière après un pré-traitement préalable selon l'équation 5.6.

$$Z_{KF3_k} = \begin{bmatrix}\dot{\theta}_{3_k}\end{bmatrix} = \underbrace{\begin{bmatrix}0 & 1\end{bmatrix}_k}_{H_{KF3}} . \underbrace{\begin{bmatrix}\theta_{KF3_k}\\ \dot{\theta}_{KF3_k}\end{bmatrix}}_{X_{KF3_k}} + v_{KF3_k} \tag{5.6}$$

Dans le pré-traitement, $\dot{\theta}_3$ est calculée à partir de la vitesse des roues arrières gauche V_{RR} et droite V_{RL}, en utilisant l'équation 5.7.

$$\dot{\theta}_3 = \frac{V_{RR} - V_{RL}}{e} \tag{5.7}$$

5.5 Services de tolérance aux fautes

Dans l'application du chapitre 3, nous avons expliqué les différents services de tolérance aux fautes proposés, et nous avons souligné qu'à cause du manque de redondance, nous pouvions détecter des fautes logicielles liées aux mécanismes de fusion de données sans pouvoir les tolérer. Dans cette seconde application de l'architecture de duplication comparaison proposée au chapitre 2, nous avons diversifié les filtres de Kalman décrits précédemment dans 5.4 afin de pouvoir tolérer de telles fautes. La détection et le rétablissement d'une faute matérielle se fait de

la même manière que dans le chapitre 3, en comparant les sorties des filtres de Kalman pour la détection, et les sorties des capteurs, et les valeurs de résidus pour le diagnostic et le rétablissement. Pour tolérer les fautes logicielles, nous exploitons la diversification logicielle et la méthode de vote majoritaire.

Dans cette section nous rappelons comment détecter et tolérer des fautes matérielles, et nous détaillons la manière de tolérer les fautes logicielles pouvant exister dans les algorithmes des filtres de Kalman.

5.5.1 Détection d'erreurs

Pour détecter la présence d'une erreur dans le système, nous comparons les deux angles de lacet θ_{KF_1} et θ_{KF_2} estimés par les deux filtres KF_1 et KF_2. La détection peut donc être exprimée par un simple pseudo-code suivant :

Algorithm 5.1: Détection d'erreur en estimation de l'angle de lacet d'un robot

$\Delta\theta_{KF_1-KF_2} = \mid \theta_{KF_2}$ - $\theta_{KF_1} \mid$
if $\Delta\theta_{KF_1-KF_2} > Thrs_\theta$ **then**
  E = {Oui} `// E : indicateur d'Erreur`
else
  E = {Non}

5.5.2 Diagnostic d'erreur

Le diagnostic d'une erreur détectée se fait en comparant les sorties des capteurs employés afin de déterminer s'il s'agit d'une erreur matérielle ou logicielle. Ceci est réalisé en comparant d'une part les vitesses V_R, et V_F (sorties des capteurs odométriques arrière R-WSS et avant F-WSS), et d'autre part $\dot{\theta}_1$,et $\dot{\theta}_2$ (sorties de $Gyro_1$ et $Gyro_2$).

- Si la sortie d'un capteur s'écarte de manière significative de son dual, le système diagnostique une erreur matérielle sur l'un de ces deux capteurs.
  - ◇ Si V_R diverge significativement de V_F alors l'erreur diagnostiquée est dans l'un des capteurs odométriques R-WSS, ou F-WSS.
  - ◇ si $\dot{\theta}_1$ diverge de $\dot{\theta}_2$ alors l'erreur diagnostiquée est une erreur gyroscopique dans $Gyro_1$ ou $Gyro_2$.

  Le capteur défectueux sera par la suite identifié en comparant les résidus des filtres de Kalman KF_1 et KF_2.

- Si les sorties des capteurs sont similaires, alors le système diagnostique une erreur logicielle dans l'un des deux filtres KF_1 ou KF_2. Le filtre défectueux sera diagnostiqué cette fois en utilisant la sortie du troisième filtre KF_3 détaillé dans la section 5.4.3 au moyen de la technique de vote majoritaire (voir section 5.5.4).

5.5.3 Recouvrement d'une erreur matérielle

Dans le cas d'une erreur matérielle, le bloc erroné est identifié en comparant les résidus des deux filtres de Kalman. Le bloc ayant le résidu le plus élevé est jugé erroné. Si aucun résidu ne dépasse une valeur significative, le système indique qu'une erreur non diagnostiquée est survenue.

Connaissant maintenant le couple $((R-WSS, F-WSS)$ ou $(Gyro_1, Gyro_2))$ contenant le capteur défaillant et la branche erronée, le système a identifié le capteur erroné, et peut être rétabli en utilisant la valeur θ de la branche sans erreur, de la même façon que dans la première application.

5.5.4 Recouvrement d'une erreur logicielle

Dans le cas de détection d'une erreur logicielle, nous assurons le rétablissement du système en comparant les sorties des filtres KF_1, KF_2 et KF_3 par la technique de vote majoritaire. Sous l'hypothèse de faute unique, le filtre de Kalman erroné aura une sortie différente des deux autres comme indiqué sur la figure 5.5. Le code 5.2 montre les étapes d'exécution de vote majoritaire implémenté.

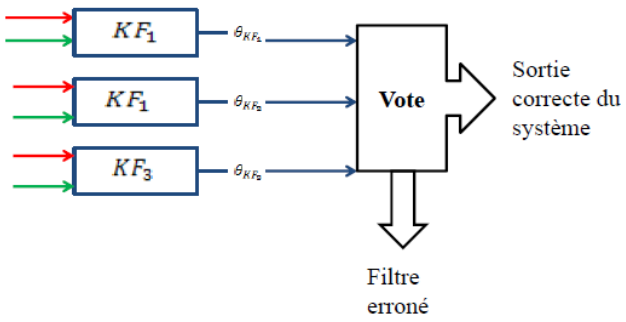

Figure 5.5 – Vote majoritaire

Algorithm 5.2: Tolérance aux fautes logicielles

Data: θ_{KF_1},θ_{KF_2},θ_{KF_3} : Sorties des filtres
Result: E : Filtre de Kalman erroné;
θ_S : Sortie du système;

Conditions :
1 :$\theta_{KF_1} \approx \theta_{KF_3} \neq \theta_{KF_2}$
2 :$\theta_{KF_2} \approx \theta_{KF_3} \neq \theta_{KF_1}$
n : entier représentant le numéro de la condition
switch n **do**
  case *1*
    $E \leftarrow \{KF_2\}$; $\theta_S \leftarrow \{\frac{\theta_{KF_1}+\theta_{KF_3}}{2}\}$;
  case *2*
    $E \leftarrow \{KF_1\}$; $\theta_S \leftarrow \{\frac{\theta_{KF_2}+\theta_{KF_3}}{2}\}$;
  otherwise
    $E \leftarrow \{KF_1, KF_2, KF_3\}$; $\theta_S \leftarrow \emptyset$;
    % *Probablement plus d'une faute dans le système*;
  end
endsw
Return(E,θ_S) ;

5.6 Évaluation des performances

Nous présentons dans cette section une évaluation des performances des mécanismes de tolérance aux fautes présentés dans ce chapitre. Pour cette expérimentation, nous utilisons exactement le même environnement matériel et logiciel que celui présenté dans le chapitre 4. Nous nous focalisons cependant sur la présentation des résultats de tolérance aux fautes logicielles et n'injectons ainsi que des fautes logicielles dans le système.

5.6.1 Relevés et Mesures

Pour l'application décrite dans ce chapitre nous avons extrait plusieurs relevés. Les données engendrées par une expérience sont sauvegardées dans des fichiers journaux dans des répertoires spécifiques à chaque type de faute injectée. A partir de ces relevés, nous avons défini les mêmes mesures que celles présentées au chapitre 4. Nous reprenons leur description dans la suite de cette section :

- *Le délai de détection* (Del_{Det_θ}) est la différence entre l'instant d'injection de la faute t_{Inj_θ} et l'instant de détection t_{Det_θ} (i.e. l'instant ou la différence entre les deux angles estimés par les deux filtres de Kalman dépasse le seuil de détection

prédéfini $Thrs_\theta$).

$$\begin{cases} Del_{Det_\theta} = t_{Det_\theta} - t_{Inj_\theta} \\ \text{Avec} \\ t_{Det_\theta} = \underset{\Delta_{\theta_{KF_1-KF_2}} \geq Thrs_\theta}{t_k} \end{cases} \tag{5.8}$$

- Les mesures B_d, B_i et B_r sont définies de la même manière que dans le chapitre 4.

Posons $(\theta_{KF_{1N}}, \theta_{KF_{2N}})$ les angles des deux filtres de Kalman KF_1, KF_2 en absence de fautes, et $(\theta_{KF_{1F}}, \theta_{KF_{2F}})$ les angles des deux filtres de Kalman KF_1, KF_2 respectivement avec une faute injectée. Du fait de l'hypothèse de faute unique, seule une branche est concernée par la faute. Dans la suite on supposera qu'il s'agit de la branche 2 (choix arbitraire puisque les deux possibilités sont symétriques). Dans ce cas $\theta_{KF_{1F}} = \theta_{KF_{1N}}$.

Sur la figure 5.6, nous définissons θ_N l'angle moyen entre les angles $\theta_{KF_{1N}}$ et $\theta_{KF_{2N}}$, θ_F l'angle moyen entre les angles $\theta_{KF_{1F}}$ et $\theta_{KF_{2F}}$ tels qu'ils sont définis dans l'équation 5.9.

$$\begin{cases} \theta_F = \frac{\theta_{KF_{1F}} + \theta_{KF_{2F}}}{2} \\ \theta_N = \frac{\theta_{KF_{1N}} + \theta_{KF_{2N}}}{2} \end{cases} \tag{5.9}$$

Nous définissons la grandeur $diff_{\theta_{CF,CN}}$ la différence entre les deux angles θ_N et θ_F suivant l'équation 5.10 :

$$diff_{\theta_{CF,CN}} = \mid \theta_F - \theta_N \mid \tag{5.10}$$

Comme indiqué dans le chapitre 4, dans nos expériences d'injection de fautes, notre mécanisme de détection ne connait que les sorties des deux filtres de Kalman avec la faute injectée : dans ce cas d'application $\theta_{KF_{1F}}$ et $\theta_{KF_{2F}}$. Cependant pour évaluer la tolérance aux fautes une comparaison de ces sorties avec celles de comportement nominal est nécessaire, ce qui est possible dans notre cas puisque les fautes sont injectées intentionnellement. Il est évident que le système ne dispose pas de cette information et doit se reposer seulement sur les valeurs de ses deux filtres de Kalman pour détecter la faute. Pour cette raison on se place encore une fois dans cette application en tant qu'observateur "omniscient" et on peut donc utiliser également les sorties des filtres dans le comportement nominal pour calculer $diff_{\theta_{CF,CN}}$.

De même que dans le chapitre 4, nous définissons encore deux variables booléennes B_{Err_θ} et B_{Def_θ} pour caractériser notre système en termes de *faux positifs*

[1] et d'*absence de détection* [2] :

- B_{Err_θ} détermine si la faute injectée engendre une erreur dans le système, sans qu'il soit forcement considéré défaillant. Nous posons ici que le système contient une erreur significative si la différence entre le comportement nominal et le comportement erroné du système est supérieure à $Thrs_{Err_\theta} = 0.1$ radian. Ainsi la variable B_{Err_θ} est définie comme suit :

$$\begin{cases} B_{Err_\theta} = 1 \text{ Si } diff_{\theta_{CF,CN}} >= (Thrs_{Err_\theta} = 0.1rad) \\ B_{Err_\theta} = 0 \text{ Si } diff_{\theta_{CF,CN}} < (Thrs_{Err_\theta} = 0.1rad) \end{cases} \tag{5.11}$$

- B_{Def_θ} détermine si la faute injectée engendre une défaillance du système. Nous considérons que le système étudié est défaillant si la différence entre le comportement nominal et le comportement avec la faute injectée est supérieure à $Thrs_{Def_\theta} = 0.5$ radian. Par conséquent cette variable B_{Def_θ} est définie comme suit :

$$\begin{cases} B_{Def_\theta} = 1 \text{ Si } diff_{\theta_{CF,CN}} >= (Thrs_{Def_\theta} = 0.5rad) \\ B_{Def_\theta} = 0 \text{ Si } diff_{\theta_{CF,CN}} < (Thrs_{Def_\theta} = 0.5rad) \end{cases} \tag{5.12}$$

Comme indiqué dans le chapitre 4, dans des applications réelles les deux seuils $Thrs_{Err_\theta}$ et $Thrs_{Def_\theta}$ sont généralement fournis par la spécification du système à développer. Pour notre cas d'étude nous définissons ces deux seuils à partir de comportement nominal du système, en étudiant la figure 5.6, et en particulier deux de ses cas limites (Figure 5.7 et 5.8) à partir d'une valeur $Thrs_\theta = 0.5$ rad.

- $Thrs_{Err_\theta}$: c'est l'écart minimum entre les comportements nominal et erroné du système pour lequel nous jugeons l'erreur comme significative (figure 5.7). Dans cette figure KF_{2F} est le plus proche possible de la valeur maximale expérimentale de KF_{2N} (0.41 radian), tout en déclenchant une détection d'erreur. La valeur théorique maximale garantissant une absence de faux positifs est donc de 0.05 radian. Nous avons décidé de prendre une valeur de $Thrs_{Err_\theta}$ de 0.1 radian, supérieure à 0.05 radian, ce qui pourra occasionner de faux positifs dans nos résultats.

- $Thrs_{Def_\theta}$: c'est l'écart minimum entre le comportement nominal et erroné pour lequel nous considérons le système comme défaillant (figure 5.8). En effet dans cette figure KF_{2F} est le plus loin possible de la valeur maximale expérimentale

[1] Faux positifs : alertes en absence d'erreurs dans le système.
[2] Absence de détection : absence d'alertes en présence d'erreurs dans le système.

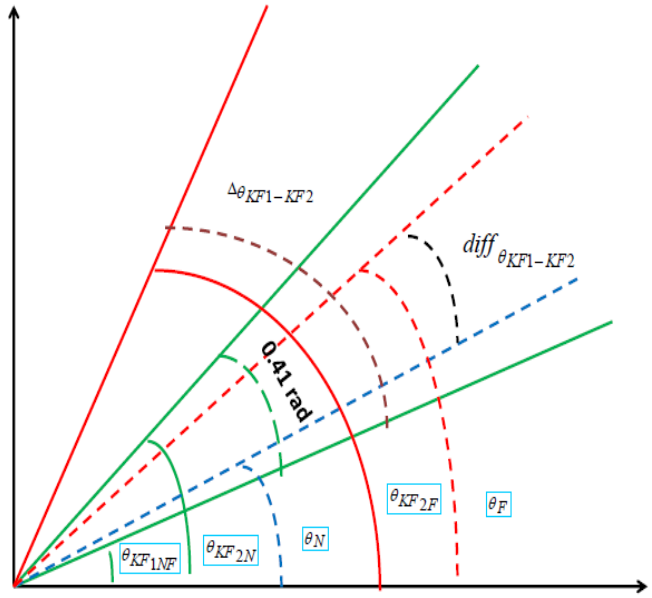

Figure 5.6 – Détermination de $Thrs_{Err_\theta}$ et $Thrs_{Def_\theta}$

de KF_{2N}, tout en ne déclenchant pas d'erreur. La valeur théorique minimale garantissant la détection est donc 0.45 radian. Avec une valeur $Thrs_{Def_\theta}$ de 0.5 radian, nous garantissons que toutes les défaillances causées par les fautes injectées seront détectées.

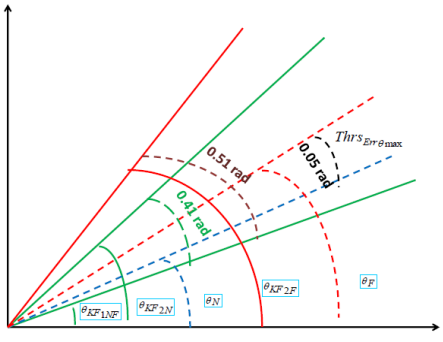

Figure 5.7 – Détermination de $Thrs_{Err_\theta}$

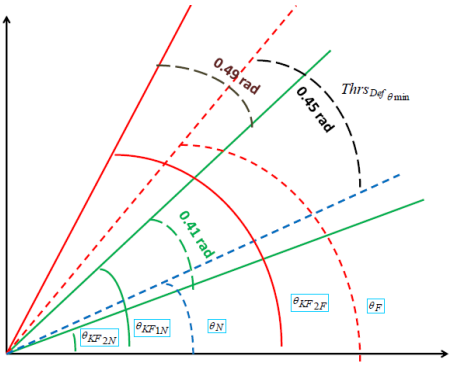

Figure 5.8 – Détermination de $Thrs_{Def_\theta}$

A partir de ces mesures, nous définissons encore pour chaque faute injectée certains pourcentages caractérisant la détection, l'identification et le recouvrement dans nos expériences :

- P_{FP_θ} est le taux de *faux positifs* [3]. Un *faux positif* est déclaré si la faute injectée est détectée comme erreur ($B_d = 1$) sans que le système soit erroné ($B_{Err_\theta} = 0$) :

$$\begin{cases} P_{FP_\theta} = \dfrac{\text{Nombre de faux positifs}}{\text{Nombre de fautes détectées}} \\ \text{avec :} \\ B_d \& \overline{B_{Err_\theta}} \quad \Rightarrow \text{faux positif} \end{cases} \tag{5.13}$$

La figure 5.9 montre un exemple de faux positif. En effet sur cette figure la valeur de $\Delta_{\theta_{KF_{1N}-KF_{2F}}}$ est égale à 0.6 radian. Elle est strictement supérieure à la valeur du seuil de détection ($Thrs_{Det_\theta} = 0.5$ radian), par conséquent notre architecture signale une détection d'erreur. Cependant, la différence entre le comportement nominal du système et son comportement avec la faute injectée (c'est-à-dire la différence entre les deux angles θ_N et θ_F $diff_{\theta_{CF,CN}}$) est égale à 0.095 radian. Elle est strictement inférieure à la valeur du seuil ($Thrs_{Err_\theta} = 0.5$ radian), par conséquent notre système détecte une erreur qui n'est pas significative que nous considérons dans notre étude comme un *faux positif*.

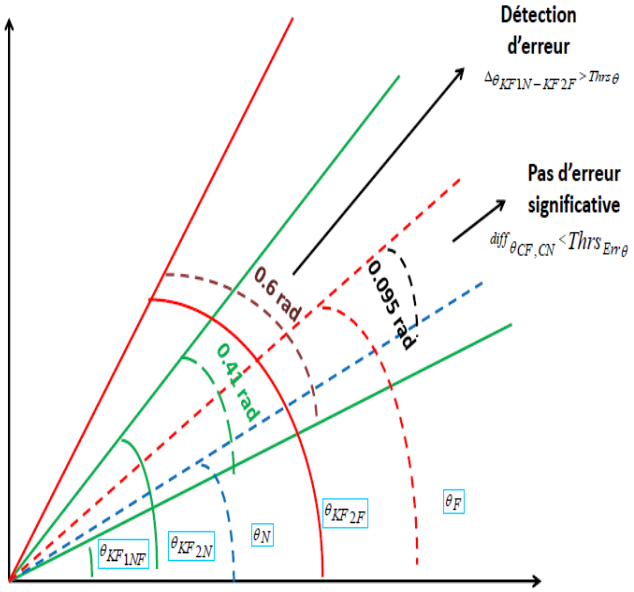

Figure 5.9 – Exemple de faux positif

- P_{ND_θ} est le taux d'*absence de détection* [4]. Une *absence de détection* est déclarée si l'erreur engendrée par la faute injectée n'est pas détectée ($B_d = 0$) sachant

[3] faux positifs : alertes en absence d'erreurs dans le système
[4] Absence de détection : absence d'alertes en présence d'erreurs dans le système

que le système est jugé défaillant ($B_{Def_\theta} = 1$) :

$$\begin{cases} P_{ND_\theta} = \dfrac{\text{Nombre de fautes non detectées}}{\text{Nombre de fautes causant une defaillance}} \\ \text{avec :} \\ \overline{B_d}\&B_{Def_\theta} \quad \Rightarrow \text{absence de détection} \end{cases} \tag{5.14}$$

A noter que selon la valeur choisie de $Thrs_{Def_\theta}$ d'après la figure 5.8, de telles absences de détection ne peuvent pas se produire.

A noter qu'il peut aussi y avoir des erreurs sans défaillance et sans détection (Figure 5.10). Dans ce cas la différence entre le comportement nominal du système et son comportement avec la faute injectée ($diff_{\theta_{CF,CN}}$) est égale à 0.55 radian. Cette valeur est strictement supérieure au seuil ($Thrs_{Err_\theta} = 0.5$ radian), ce qui signifie qu'une erreur significative est présente dans le système. Cependant la distance $\Delta_{\theta_{KF_1-KF_2}}$ est égale à 0.4 radian. Elle est strictement inférieure au seuil de détection choisi ($Thrs_{Det_\theta} = 0.5$ radian), ce qui implique qu'il n'y a pas de détection. Ce comportement est correct d'après nos définitions : bien qu'il y ait une erreur, il n' y a pas de défaillance et le système n'a pas d'obligation à soulever une détection d'erreur.

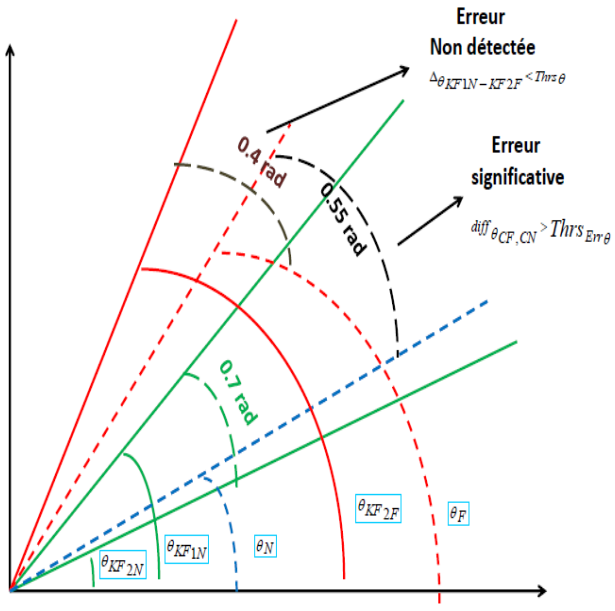

Figure 5.10 – Exemple d'erreur non détectée

- P_i et P_r sont les pourcentages de fautes détectées correctement diagnostiquées et rétablies comme définies au chapitre 4.

- P_{ESD_θ} : est le pourcentage de fautes détectées avec comportement erroné sans comportement défaillant du système : C'est le taux de fautes pour lesquelles l'expression logique $B_d \& B_{Err_\theta} \& \overline{B_{Def_\theta}}$ est vraie. Pour ces fautes la détection est correcte : il y a bien une erreur mais cette erreur n'a pas causé de défaillance. De telles détections pourraient à la rigueur être considérées comme des détections intempestives. Comme précisé au chapitre précédent, ce sera au concepteur de décider quel comportement du système définir face à ces fautes.

De la même manière que dans le chapitre 4 nous avons défini trois mesures temporelles :

- Del_{Def_θ} détermine au bout de combien de temps la faute injectée cause une défaillance du système :

$$\begin{cases} Del_{Def_\theta} = t_{Def_\theta} - t_{Inj_\theta} \\ \text{Avec} \\ t_{Def_\theta} = \underset{Diff_{\theta_{(CF,CN)}} \geq Thrs_{Def_\theta}}{t_k} \end{cases} \tag{5.15}$$

- Del_{Err_θ} détermine au bout de combien de temps une erreur significative est détectée.

$$\begin{cases} Del_{Err_\theta} = t_{Det_\theta} - t_{Err_\theta} \\ \text{Avec} \\ t_{Err_\theta} = \underset{Diff_{\theta_{(CF,CN)}} \geq Thrs_{Err_\theta}}{t_k} \end{cases} \tag{5.16}$$

- $Del_{\theta_{Def,Det}}$ est le temps restant après détection jusqu'à la défaillance.

$$\begin{cases} Del_{\theta_{Def,Det}} = t_{Def_\theta} - t_{Det_\theta} \\ \text{Avec} \\ t_{Det_\theta} = \underset{\Delta_{\theta_{KF_1-KF_2}} \geq Thrs_{Det_\theta}}{t_k} \end{cases} \tag{5.17}$$

La table 5.1 résume les différentes mesures employées dans notre expérimentation :

Mesure	Désignation	Unité
Del_{Det_θ}	Délai de détection	Seconde
B_d	Indicateur de détection	Booléen
B_i	Indicateur d'identification	Booléen
B_r	Indicateur de rétablissement	Booléen
B_{Err_θ}	Indicateur d'erreur	Booléen
B_{Def_θ}	Indicateur de défaillance	Booléen
P_{FP_θ}	Taux de faux positifs	%
P_{ND_θ}	Taux d'absence de détection	%
P_i	Taux d'identification	%
P_r	Taux de rétablissement	%
P_{ESD_θ}	Taux d'erreur sans défaillance	%
Del_{Def_θ}	Délai de défaillance	Seconde
Del_{Err_θ}	Délai d'erreur	Seconde
$Del_{\theta_{Def,Det}}$	Délai entre la détection et la défaillance	Seconde

Tableau 5.1 – Résumé des mesures pour le cas d'application d'estimation d'angle de lacet d'un robot mobile

5.6.2 Fautes injectées

Afin d'évaluer la performance et l'efficacité des mécanismes de tolérance aux fautes proposés dans ce chapitre, nous injectons des fautes par mutation directement dans le modèle du processus ou d'observation des filtres de Kalman. Dans cette évaluation, nous nous basons sur deux type de mutations :

- Substitution de valeurs numériques
- Substitution d'expressions géométriques

Nous avons injecté huit fautes dans la première branche de notre architecture (KF_1), et six dans KF_2. Au total nous avons simulé quatorze fautes. Ces dernières sont représentées dans le tableau 5.2.

Bloc	Modèle	Éléments matriciel	Symboles	Mutation	
				Original	Mutant
KF_1	Modèle du processus	$\underbrace{\begin{bmatrix}1 & 0\\ 0 & 0\end{bmatrix}}_{A_{KF_1}}$	$\underbrace{\begin{bmatrix}A^1_{1,1} & A^1_{1,2}\\ A^1_{2,1} & A^1_{2,2}\end{bmatrix}}_{A_{KF_1}}$	$A^1_{1,1} : 1$ $A^1_{1,1} : 1$	$A^1_{1,1} : 0.99$ $A^1_{1,1} : -0.99$
KF_1	Modèle du processus	$\underbrace{\begin{bmatrix}\frac{\Delta t}{e} & \frac{-\Delta t}{e}\\ \frac{1}{e} & -\frac{1}{e}\end{bmatrix}}_{B_{KF_1}}$	$\underbrace{\begin{bmatrix}B^1_{1,1} & B^1_{1,2}\\ B^1_{2,1} & B^1_{2,2}\end{bmatrix}}_{B_{KF_1}}$	$B^1_{1,1} : \frac{\Delta t}{e}$ $B^1_{1,1} : \frac{\Delta t}{e}$ $B^1_{1,2} : \frac{-\Delta t}{e}$	$B^1_{1,1} : \frac{\Delta t}{2*e}$ $B^1_{1,1} : \frac{2*\Delta t}{e}$ $B^1_{1,2} : \frac{\Delta t}{e}$
KF_1	Modèle d'observation	$\underbrace{\begin{bmatrix}0 & 1\end{bmatrix}}_{H_{KF_1}}$	$\underbrace{\begin{bmatrix}H^1_{1,1} & H^1_{1,2}\end{bmatrix}}_{H_{KF_1}}$	$H^1_{1,1} : 0$ $H^1_{1,1} : 0$	$H^1_{1,1} : \frac{1}{100}$ $H^1_{1,1} : -\frac{1}{100}$
KF_2	Modèle du processus	$\underbrace{\begin{bmatrix}1 & 0\\ 0 & 0\end{bmatrix}}_{A_{KF_2}}$	$\underbrace{\begin{bmatrix}A^2_{1,1} & A^2_{1,2}\\ A^2_{2,1} & A^2_{2,2}\end{bmatrix}}_{A_{KF_2}}$	$A^2_{1,1} : 1$ $A^2_{1,1} : 1$	$A^2_{1,1} : 0.99$ $A^2_{1,1} : -0.99$
KF_2	Modèle du processus	$\underbrace{\begin{bmatrix}\frac{\Delta t sin\phi}{L}\\ \frac{sin\phi}{L}\end{bmatrix}}_{B_{KF_2}}$	$\underbrace{\begin{bmatrix}B^2_{1,1}\\ B^2_{2,1}\end{bmatrix}}_{B_{KF_2}}$	$B^2_{1,1} : \frac{\Delta t sin\phi}{L}$ $B^2_{1,1} : \frac{\Delta t sin\phi}{L}$ $B^2_{1,1} : \frac{\Delta t sin\phi}{L}$	$B^2_{1,1} : \frac{\Delta t cos\phi}{L}$ $B^2_{1,1} : -\frac{\Delta t sin\phi}{L}$ $B^2_{1,1} : \frac{\Delta t tg\phi}{L}$
KF_2	Modèle d'observation	$\underbrace{\begin{bmatrix}0 & 1\end{bmatrix}}_{H_{KF_2}}$	$\underbrace{\begin{bmatrix}H^1_{1,1} & H^1_{1,2}\end{bmatrix}}_{H_{KF_2}}$	$H^2_{1,1} : 0$ $H^2_{1,1} : 0$	$H^2_{1,1} : \frac{1}{100}$ $H^2_{1,1} : -\frac{1}{100}$

Tableau 5.2 – Fautes logicielles injectées

5.6.3 Résultats généraux

Nous présentons dans cette section les résultats que nous avons obtenus dans notre campagne d'expérimentation pour ce cas d'application. Nous présentons d'abord un exemple d'expérimentation montrant le comportement nominal du système et un autre exemple montrant l'effet d'une faute logicielle injectée. Enfin nous présentons les mesures réalisées pendant notre campagne d'injections de fautes.

5.6.3.1 Exemple d'expérimentation

Pour évaluer la détection et le recouvrement de fautes logicielles des mécanismes proposés dans ce chapitre, nous avons utilisé l'environnement logiciel détaillé dans le chapitre 4. Le véhicule parcourt le circuit de test représenté dans la figure 4.1, et les différents composants logiciels de la plateforme PACPUS enregistrent les données perçus par les capteurs :

- Les capteurs odométrique R-WSS, et F-WSS délivrent les vitesses V_{RR}, V_{RL}, et V_F respectivement avec une fréquence de 25 Hz.
- les deux Gyroscopes $Gyro_1$, $Gyro_2$ fournissent les vitesses angulaires $\dot{\theta}_1$ et $\dot{\theta}_2$ respectivement avec une fréquence de 100Hz.

Ces différentes données sont rejouées et exportées dans des matrices Matlab, et synchronisées par une interpolation linéaire.

Dans ce qui suit nous présentons deux exemples d'expérimentation. Le premier présente le comportement nominal du système, et le second son comportement avec une faute logicielle injectée pour montrer la façon de détecter et de tolérer les fautes logicielles. Nous considérons comme faute une mutation réalisée dans le modèle du processus de filtre de Kalman KF_1 : nous remplaçons dans la matrice de commande B de modèle du processus du premier filtre KF_1 la constante e par $2e$.

- **Comportement nominal sans faute injectée**

  Dans un premier temps nous présentons le comportement nominal du système. Cette expérimentation est nécessaire car elle nous permet de fixer les différents seuils employés dans notre démarche de tolérance aux fautes. Pour la détection de fautes, nous avons choisi un seuil de détection $Thrs_\theta = 0.5$ radian supérieur à la différence $\Delta\theta_{KF_1-KF_2}$ comme indiqué dans la figure 5.11.

  La Figure 5.12 représente l'angle de lacet du véhicule estimé par les deux filtres de Kalman $(\theta_{KF_1}, \theta_{KF_2})$ et la sortie de notre architecture θ_S, qui est la moyenne de ces deux estimations.

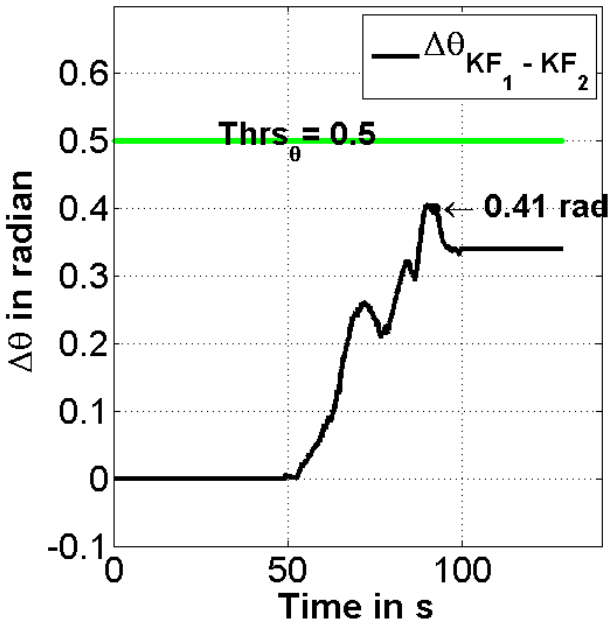

Figure 5.11 – Comportement nominal : Différence entre les angles estimés par les deux filtres de Kalman

Trois autres seuils sont aussi nécessaires dans le cadre de l'application présentée dans ce chapitre :

◇ Un seuil pour comparer les sorties des deux encodeurs R-WSS et F-WSS : $Thrs_V = 0.26$ mètres/seconde.

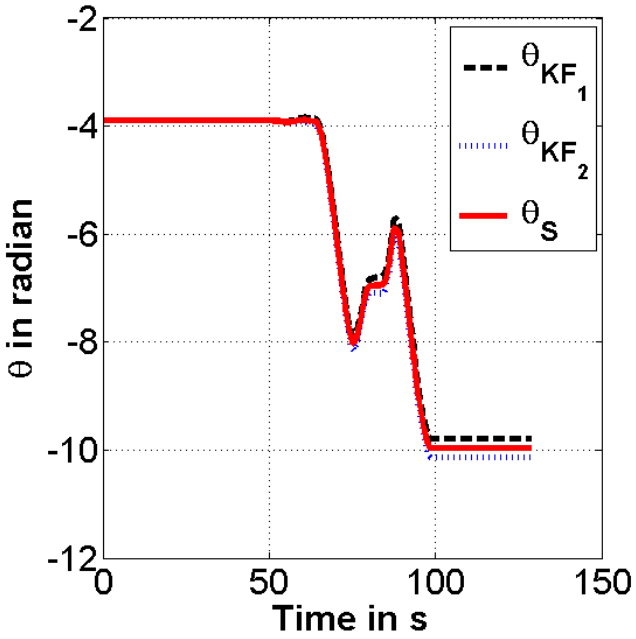

Figure 5.12 – Comportement nominal : L'angle θ estimé par les deux filtres de Kalman et le système

◇ Un seuil pour comparer les sorties des deux gyromètres $Gyro_1$, $Gyro_2$: $Thrs_{\dot{\theta}} = 0.27$ radian/seconde.

◇ Un seuil pour s'assurer, en cas d'erreur matérielle détectée, que le conflit dans un filtre de Kalman est considérablement plus élevé que dans l'autre $Thrs_{Res} = 0.15$ radian.

La table 5.3 résume les différents seuils employés dans notre campagne d'expérimentation.

- **Comportement du système en présence de faute**

Dans cette expérience, nous avons injecté une faute logicielle dans le filtre de Kalman KF_1 à l'instant $t_{Inj_\theta} = 0$ seconde, nous avons remplacé la valeur de premier élément (initialement égale à $\frac{\Delta t}{e}$) de la matrice de commande B par la valeur $\frac{\Delta t}{2e}$. Cette exemple de mutation est représenté dans l'équation 5.18.

$$\underbrace{\begin{bmatrix} \frac{\Delta t}{e} & \frac{-\Delta t}{e} \\ \frac{1}{e} & -\frac{1}{e} \end{bmatrix}}_{B_{KF_1}} \underset{Mutation}{\Longrightarrow} \underbrace{\begin{bmatrix} \frac{\Delta t}{2e} & \frac{-\Delta t}{e} \\ \frac{1}{e} & -\frac{1}{e} \end{bmatrix}}_{B_{KF_1}} \tag{5.18}$$

La figure 5.13 montre l'effet de cette faute sur la sortie de filtre de Kalman KF_1. Comme on le voit sur cette figure, l'angle de lacet θ_{KF1} diverge sensiblement de θ_{KF2} estimé par le filtre KF_2 à l'instant $t_{Det_\theta} = 53.07$ seconde. En effet avant cet instant

Seuils	**Désignation**	**Valeur**	**Unité**
$Thrs_{\theta}$	Seuil de détection	0.5	Radian
$Thrs_{V}$	Seuil de comparaison des sorties de R-WSS et F-WSS	0.26	Mètres/Seconde
$Thrs_{\dot{\theta}}$	Seuil de comparaison des sorties des $Gyro_1$ et $Gyro_2$	0.27	$Radian/Seconde$
$Thrs_{Res}$	Seuil de comparaison des résidus	0.15	Radian
$Thrs_{Err_{\theta}}$	écart minimum entre les comportements nominal et erroné du système pour lequel nous jugeons l'erreur comme significative	0.1	Radian
$Thrs_{Def_{\theta}}$	écart minimum entre le comportement nominal et erroné pour lequel nous considérons le système comme défaillant	0.5	Radian

Tableau 5.3 – Résumé des seuils pour le cas d'application d'estimation de l'angle de lacet d'un robot mobile

le véhicule parcourt la ligne droite du circuit de test. La commande en rotation est donc proche de zéro, et l'effet de la mutation (un scalaire de cette commande) affecte peu le comportement du système. L'incidence de la mutation se concrétise par contre dès que le véhicule commence à tourner.

D'après les figures 5.14 et 5.15, nous constatons que l'erreur détectée n'est pas une erreur matérielle car la différence entre les sorties des capteurs de même type ne dépasse pas leur seuil, indiquant ainsi une faute logicielle dans l'un des algorithmes de fusion de données par filtrage de Kalman (KF_1 ou KF_2) sans être en mesure de distinguer lequel.

Pour identifier le filtre de Kalman erroné, nous comparons les sorties des deux filtres KF_1, KF_2 avec la sortie du troisième filtre diversifié KF_3 en utilisant la technique de vote majoritaire. Le filtre défaillant aura une sortie différente de celle des deux autres, et la sortie du système sera la moyenne des sorties des filtres sans erreurs comme montré sur la figure 5.16.

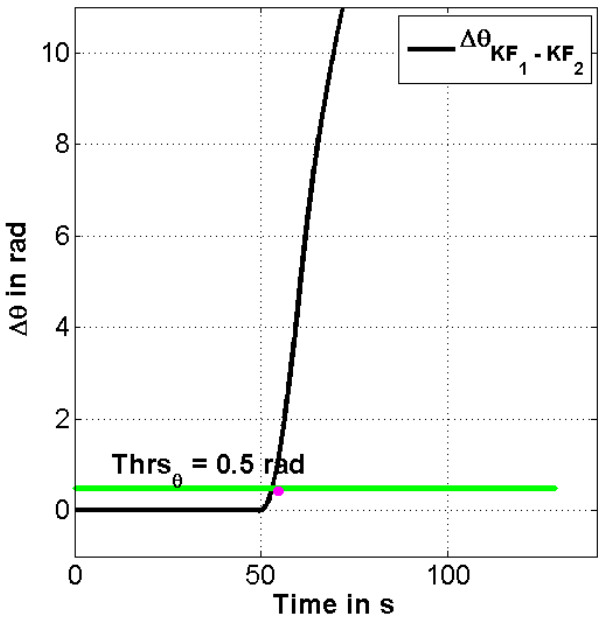

Figure 5.13 – Faute logicielle sur le filtre de Kalman 1 : Différence Δ_θ entre les angles des deux filtres de Kalman

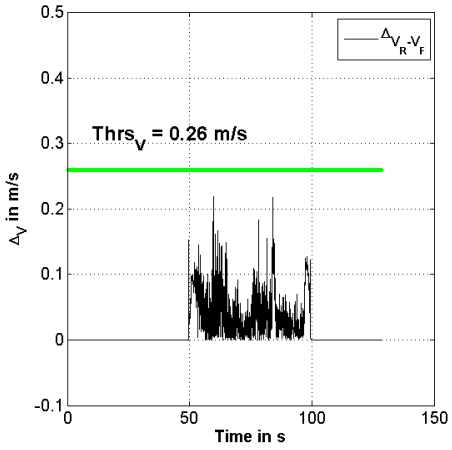

Figure 5.14 – Faute logicielle sur le filtre de Kalman 1 : Comparaison des sorties de R-WSS et F-WSS

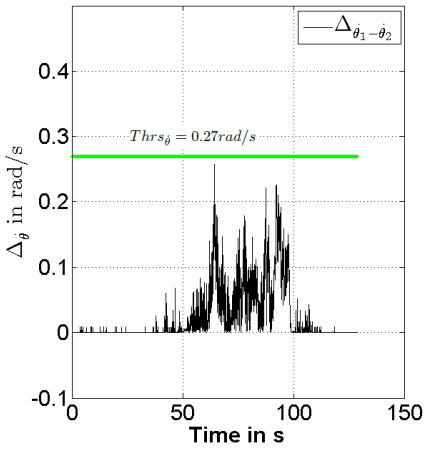

Figure 5.15 – Faute logicielle sur le filtre de Kalman 1 : Comparaison des sorties de $Gyro_1$ et $Gyro_2$

5.6.3.2 Mesures globales

Les mesures relevées lors de notre expérimentation sont représentées dans le tableau 5.4. D'après ces traces d'exécution, les 14 mutations que nous avons simulées ont une incidence sur le système ($B_{Err_\theta} = 1$), et ont été détectées ($B_d = 1$) par notre architecture. Étant donné que nous avons ajouté un niveau de redondance en diversifiant le 3^{me} filtre de Kalman KF_3, ces fautes logicielles détectées sont

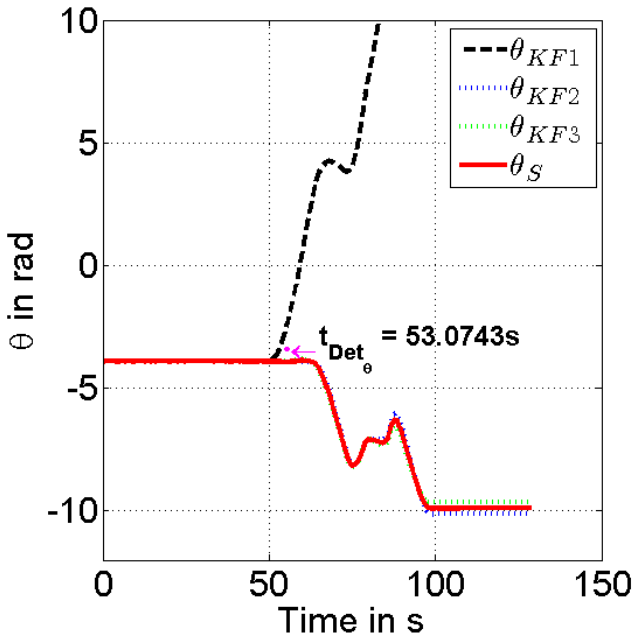

Figure 5.16 – Faute logicielle sur le filtre de Kalman 1 : L'angle estimée par les deux filtres de Kalman et le système

toutes rétablies par la méthode de vote majoritaire. Pour ces fautes logicielles, aucun *faux positif* n'est engendré et aucune *absence de détection* n'est signalée. 10% de mutations réalisées engendrent une erreur dans le système sans sa défaillance. Ces erreurs pourraient être considérées comme des détections intempestives comme mentionné dans le chapitre précédent.

Original	Mutant	t_{Inj_θ}	t_{Det_θ}	Del_{Det_θ}	Del_{Def_θ}	$Del_{(Def,Det)_\theta}$	Del_{Err_θ}	B_d	B_i	B_r	B_{Err_θ}	B_{Def_θ}
$A^1_{1,1} : 1$	$A^1_{1,1} : 0.99$	0.00	0.40	0.40	1.04	0.64	0.32	1	0	1	1	1
$A^1_{1,1} : 1$	$A^1_{1,1} : -0.99$	0.00	0.01	0.01	0.01	0.00	0.00	1	0	1	1	1
$B^1_{1,1} : \frac{\Delta t}{e}$	$B^1_{1,1} : \frac{\Delta t}{2*e}$	0.00	53.07	53.07	54.43	1.36	1.16	1	0	1	1	1
$B^1_{1,1} : \frac{\Delta t}{e}$	$B^1_{1,1} : \frac{2*\Delta t}{e}$	0.00	50.95	50.95	51.47	0.52	0.48	1	0	1	1	1
$B^1_{1,2} : \frac{-\Delta t}{e}$	$B^1_{1,2} : \frac{\Delta t}{e}$	0.00	50.59	50.59	50.95	0.36	0.44	1	0	1	1	1
$H^1_{1,1} : 0$	$H^1_{1,1} : \frac{1}{100}$	0.00	2.84	2.84	4.40	1.56	1.20	1	0	1	1	1
$H^1_{1,1} : 0$	$H^1_{1,1} : -\frac{1}{100}$	0.00	2.84	2.84	4.36	1.52	1.16	1	0	1	1	1
$A^2_{1,1} : 1$	$A^2_{1,1} : 0.99$	0.00	0.44	0.44	1.08	0.64	0.32	1	0	1	1	1
$A^2_{1,1} : 1$	$A^2_{1,1} : -0.99$	0.00	0.01	0.01	0.01	0.00	0.00	1	0	1	1	1
$B^2_{1,1} : sin$	$B^2_{1,1} : cos$	0.00	51.47	51.47	52.11	0.64	0.64	1	0	1	1	1
$B^2_{1,1} : sin$	$B^2_{1,1} : -sin$	0.00	65.51	65.51	66.07	0.56	1.28	1	0	1	1	1
$B^2_{1,1} : sin$	$B^2_{1,1} : tg$	0.00	89.95	89.95	X	X	-3.24	1	0	1	1	0
$H^2_{1,1} : 0$	$H^2_{1,1} : \frac{1}{100}$	0.00	2.84	2.84	4.36	1.52	1.20	1	0	1	1	1
$H^2_{1,1} : 0$	$H^2_{1,1} : -\frac{1}{100}$	0.00	2.84	2.84	4.36	1.52	1.20	1	0	1	1	1

Tableau 5.4 – Injection de fautes logicielles dans le cas d'estimation d'angle de lacet d'un robot

5.7 Conclusion

Nous avons présenté dans ce chapitre un exemple d'application pour l'estimation de l'angle de lacet des robots mobiles de l'architecture de *duplication-comparaison* présentée au chapitre 2.

Comme il a été présenté, nous nous sommes focalisés dans ce chapitre sur la tolérance aux fautes logicielles. Ces fautes peuvent apparaître dans les modèles des filtres de Kalman. Nous avons montré qu'en utilisant la technique de programmation 3-versions, en plus des fautes matérielles, nous tolérons aussi des fautes logicielles liées aux mécanismes de fusion de données. Nous avons diversifié trois filtres de Kalman indépendants et nous avons montré comment tolérer une faute logicielle dans les deux filtres actifs dans le système. Cette tolérance est assurée par la méthode de vote majoritaire.

Nous avons évalué les performances de nos mécanismes dans le même environnement que celui présenté au chapitre 4 (acquisition de données avec PACPUS, rejeu de données et injection de fautes avec Matlab). Les résultats d'expérimentation montrent l'efficacité du rétablissement de fautes logicielles, mécanisme qui n'était pas présent dans l'application précédente.

Conclusion

Sommaire

Dans le cadre de cette thèse, nous nous sommes intéressés aux problèmes de sûreté de fonctionnement dans les systèmes de perception multi-capteurs, en considérant en particulier les aspects de tolérance aux fautes dans la fusion de données.

Nous concluons dans ce chapitre notre manuscrit en présentant un bilan résumant les différents aspects touchés et en rappelant nos principaux résultats. Nous présentons enfin plusieurs perspectives possibles dans la continuité de nos travaux.

6.1 Bilan

Nous avons cherché dans ce manuscrit à étudier de quelles façons les techniques de tolérance aux fautes peuvent être appliquées aux systèmes de perception multi-capteurs. Notre recherche s'est trouvée motivée par la difficulté d'utiliser des approches formelles, et l'impossibilité de faire des tests exhaustifs pour valider les mécanismes de fusion de données. Ces mécanismes de fusion de données sont l'approche la plus utilisée en perception multicapteurs pour réduire les incertitudes et les imprécisions des capteurs, et améliorer la prise de décision. Cependant ils utilisent des algorithmes déclaratifs qui rendent difficiles la vérification de leur comportement par des approches formelles telles que la preuve mathématique et le model checking. De plus l'environnement ouvert et changeant auquel sont confrontés les systèmes robotiques sur le terrain engendre un contexte d'exécution quasi infini ce qui rend la tache de test couteûse et non exhaustive. Ces deux limitations engendrent

l'impossibilité d'éliminer toutes les fautes dans ces systèmes de perception et les rend incapables de donner une confiance justifiée dans la sécurité-innocuité et la fiabilité de leur comportement. Pour remédier à ces contraintes de validation, nous avons décidé d'utiliser une solution alternative à l'élimination de fautes en implémentant des mécanismes de tolérance aux fautes dans les systèmes de perception, permettant de délivrer un service correct d'une manière continue malgré la présence de fautes.

Le premier chapitre de ce document a permis d'effectuer un large tour d'horizon des deux domaines scientifiques traités dans ces travaux de recherche, à savoir la sûreté de fonctionnement (particulièrement les méthodes de tolérance aux fautes), et la fusion de données multicapteurs (particulièrement les méthodes basées sur les fonctions de croyance et celles basées sur le filtrage de Kalman). Une étude critique des techniques de tolérance aux fautes en fusion de données a été réalisée. Nous avons constaté que les méthodes proposées dans la littérature consistent à utiliser les mécanismes de fusion de données pour tolérer des fautes matérielles sur les capteurs. Or pour nous ces mécanismes eux mêmes ne sont pas dignes de confiance. Nous avons par conséquent particulièrement cherché à traiter cet aspect dans les travaux proposés.

Le second chapitre présente ainsi notre réponse aux manques et aux critiques faites sur les méthodes proposées dans la littérature. Nous avons proposé deux architectures différentes : la première utilise la technique de duplication-comparaison, la seconde est totalement basée sur la fusion de données avec une analyse de son comportement pour garantir son bon fonctionnement. Ce chapitre détaille dans un premier temps comment les mécanismes de détection, de diagnostic et de recouvrement sont pris en compte dans les deux approches proposées, et dans un second temps, il donne une analyse et une comparaison qualitatives de ces deux approches.

Le troisième chapitre présente le contexte applicatif de l'approche de duplication-comparaison proposée. Il détaille dans un premier temps les mécanismes de fusion de données utilisés (les filtres de Kalman) et leurs implémentation pour la localisation des robots mobiles. Dans un second temps il précise les différents services de tolérance aux fautes offerts pour cette application à savoir : la détection de fautes matérielles dans les capteurs et logicielles dans les filtre de Kalman, et le rétablissement du système

Le quatrième chapitre a détaillé le contexte expérimental en évaluant les performances de notre étude de cas présentée dans le chapitre 3. Un environnement de validation basé sur le rejeu de données et l'injection de fautes a été mis en place, et les différents résultats montrent l'efficacité de l'approche proposée face aux différentes fautes injectées. En dépit de ces fautes qui simulent des situations

réelles que peut rencontrer un véhicule dans son environnement, notre approche a été capable d'offrir des services de détection de fautes et de rétablissement au système.

Le cinquième chapitre est consacré à la tolérance aux fautes logicielles. Il présente une application d'estimation de l'angle de lacet d'un robot mobile utilisant le filtre de Kalman. Il emploie la diversification fonctionnelle par la programmation N-versions pour assurer la détection et le recouvrement d'une faute de conception et de développement pouvant exister dans les algorithmes de filtre de Kalman. Une validation des mécanismes proposés est détaillée.

6.2 Leçon à retenir

La perception est considérée comme une entrée fondamentale des systèmes robotiques. En effet ces systèmes prennent leurs décisions et commandent leurs actions suite a cette perception, qui est donc une fonction critique et doit être correcte et fiable pour éviter des conséquences catastrophiques sur le robot et son environnement. Étant donné que les capteurs de perception généralement utilisés en robotique mobile sont imprécis et incertains, des mécanismes de fusion de données sont développés pour corriger ces défauts. Cependant ces mécanismes d'intelligence artificielle sont difficiles a valider et évaluer. D'une part leurs programmation déclarative rend complexe voire impossible l'analyse de leur comportement par une vérification formelle. D'autre part la vaste dimension de contextes d'exécution des systèmes robotiques rend difficile et couteuse l'application des techniques d'élimination des fautes (comme le test). Intégrer dans ces mécanismes de fusion de données des techniques de tolérance aux fautes (comme le doublement par comparaison ou le masquage de fautes) nous parait donc fortement recommandé pour assurer leurs bon fonctionnement et donner une confiance justifiée dans le service qu'ils délivrent.

Les mécanismes proposés dans ces travaux, bien qu'améliorant objectivement le comportement des systèmes de perception multi-capteurs en présence de fautes, ne répondent qu'à l'aspect de la tolérance aux fautes internes au système. En effet, ils se concentrent sur les fautes physiques qui peuvent survenir sur les capteurs (sources de données), et sur les fautes logicielles qui peuvent êtres engendrées dans les algorithmes de fusion de données. Ils sont ainsi complémentaires à la fusion de données qui couvre d'autres types de fautes : les fautes externes qu'elles soient dues à des perturbations extérieures ou à des aléas de l'environnement. Il est aussi important de noter que le travail d'expérimentation évaluant les mécanismes proposés ne porte que sur un exemple d'application de localisation des robots mobiles. Des tests sur d'autres types d'application utilisant la fusion de données

tels que la détection d'obstacles, le suivi d'objet, ou l'association de données sont nécessaires pour s'assurer du bon fonctionnement des mécanismes proposés dans des cas aussi nombreux et variés que possible. Néanmoins, les travaux menés dans ce manuscrit montrent objectivement que des mécanismes de tolérance aux fautes appliqués à la fusion de données améliorent sensiblement le comportement d'un système de perception multi-capteurs en présence de fautes. Ils montrent en particulier que des fautes logicielles dans les mécanismes de fusion peuvent être difficiles à éliminer et doivent être pris en considération pour la sureté de fonctionnement du système. Nous considérons donc souhaitable l'utilisation de tels mécanismes dans les applications critiques.

L'environnement d'évaluation des mécanismes de tolérance aux fautes mis en place s'est avéré en adéquation avec la réalité, et apte à mesurer l'impact des fautes injectées sur le comportement du système de perception développé. Basée sur l'acquisition réelle, et le rejeu de donnée sur Matlab, cette approche ne doit pas se substituer aux tests directement dans l'environnement réel, mais doit permettre, à travers la campagne d'injection de fautes d'effectuer des tests intensifs et une évaluation efficace en termes de sureté de fonctionnement. Ce principe de rejeu de données et d'injection de fautes est largement utilisé pour simuler les différentes situations adverses qu'un système de perception ou le système robotique dans son ensemble peut rencontrer dans son environnement. De plus les fautes sont injectées de différentes façons a des instant différents sur le système, de manière contrôlée et sans danger afin de révéler leurs conséquences sur le comportement du système. Notons enfin que le travail nécessaire pour mettre en œuvre un environnement d'évaluation ne doit pas être sous-estimé, mais il est cependant nécessaire pour l'évaluation de la sûreté de fonctionnement du système.

Pour illustrer ce dernier argument, nous avons passé plusieurs mois à déplacer une application de localisation sur des robots mobiles Wifibot de petite dimension, sans aboutir à un résultat satisfaisant. En effet, les GPS utilisés n'étaient pas assez précis, et les accéléromètres souffraient des vibrations du robot (sans composant atténuateur, comme les suspensions de voiture).

6.3 Principales Contributions

Nos principales contributions dans ces travaux de recherche peuvent êtres résumées dans cette liste :

- Étude théorique des méthodes de tolérance aux fautes en fusion de données proposées dans la littérature et leur critique : les mécanismes proposés dans

la littérature se concentrent principalement sur les fautes matérielles, en s'appuyant directement sur les mécanismes de fusion pour détecter et tolérer les erreurs. Mais les mécanismes de fusion sont difficiles à concevoir et valider et souvent intègrent des valeurs qui sont déterminées empiriquement (comme les gains) ou des modèles qui sont difficiles à valider formellement. Les défaillances logicielles dans les algorithmes de fusion sont donc pour nous une grande préoccupation et les mécanismes de tolérance aux fautes à développer doivent tolérer à la fois les fautes matérielles dans les sources de données et logicielles dans les algorithmes de fusion de données.

- Étude formelle qualitative d'un mécanisme de fusion de données utilisant les fonctions de croyance. Cette étude simplifiée est une preuve de concept qu'une analyse formelle des mécanismes de fusion de données est possible. Cependant, dans la pratique, une telle analyse peut se heurter rapidement à une explosion combinatoire due aux domaines de variables réelles ou aux facteurs temporels.

- Proposition d'une architecture générique de duplication- comparaison assurant des services de tolérance aux fautes en fusion de données qui :

  ◇ détecte et tolère une faute matérielle dans les sources de données

  ◇ détecte et tolère une faute logicielle dans les mécanismes de fusion de données ; cette tolérance est assurée par la diversification logicielle et la méthode de vote.

- Application de l'architecture proposée pour la localisation en robotique mobile, et démonstration de son applicabilité pour assurer la détection de faute et le rétablissement du système dans la fusion de données par filtrage de Kalman. Mise en place d'un environnement d'évaluation de tolérance aux fautes. Cet environnement est basé sur le rejeu de données et l'injection de fautes.

- Application de l'architecture proposée pour l'estimation de l'angle de lacet d'un robot mobile, et utilisation de la diversification fonctionnelle pour assurer le rétablissement du système suite à une faute logicielle. Évaluation des performances des mécanismes proposés en termes de détection et recouvrement.

6.4 Perspectives

Nous présentons dans cette section les différentes perspectives et pistes de recherche pouvant améliorer ces travaux de thèses à court et long terme

6.4.1 Perspectives a court terme

Pour enrichir ces travaux, des améliorations sans modifications significatives des approches proposées sont possibles. Ainsi à court terme nous envisageons :

- d'automatiser davantage l'injection de fautes pour considérer un plus grand nombre de fautes pour continuer la validation des architectures proposées, en particulier par la génération automatique de fautes à injecter.
- d'éliminer l'hypothèse de faute unique en ajoutant plus de redondance, à la fois matérielle et logicielle.
- d'analyser formellement le modèle du filtre de Kalman pour comparer sa tolérance aux fautes intrinsèque à celle de notre architecture,
- de tester notre approche dans le cas de fautes transitoires, où nous supposons que par l'entretien ou le temps, une faute activée peut disparaître.
- de réaliser une étude comparative des résultats obtenus dans notre campagne d'expérimentation suivant des seuils de détection différents, cette étude nous permettra de bien choisir un seuil optimal.

6.4.2 Perspectives à long terme

Les travaux de thèse présentés dans ce manuscrit ouvrent plusieurs pistes de recherche, donc a plus long terme nous envisageons :

- de faire une étude comparative en termes de tolérance aux fautes entre les différentes méthodes de filtrage (EKF, UKF, filtre particulaire ...).
- d'appliquer l'architecture générique de duplication-comparaison présentée dans le chapitre 2 dans un autre type d'application dans le domaine de l'automobile, telle que la détection d'obstacle, ou le suivi d'objet.
- d'essayer de modéliser un problème de perception selon les trois formalismes présentés dans le premier chapitre (probabiliste, possibiliste, et evidentiel). Une telle modélisation consiste fondamentalement en une diversification, et pourrait être exploitée pour assurer des services de tolérance aux fautes.
- préconiser les travaux d'injection de fautes en développant des fonctions compatibles avec la plateforme PACPUS et réutilisable pour de futurs travaux de validation sur d'autres applications.

Résultats de notre experimentation

Sommaire

Nous présentons ici les différents résultats de toute notre expérimentation sous forme de tables dont quelques échantillons sont déjà détaillés au chapitre 4. Pour chaque composant de notre architecture nous présentons les différentes mesures discutées auparavant au chapitre 4. Nous détaillons les différents scénarios issus de la campagne d'injection de fautes selon le type de fautes injectée.

A.1 Fautes injectées sur le GPS

Les quatre tables A.1, A.2, A.3 et A.4 qui suivent représentent les résultats de notre campagne d'injection sur le GPS. 64 scénarios sont testés dans ce cas (quatre sauts (2,5,10, 20) sont ajoutés aux sorties du GPS à des instants différents suivants les quatre directions possibles (+X,-X,+Y, -Y)).

Comp	Saut	Dir	t_{Inj}	t_{Det}	Del_{Det}	Del_{Def}	$Del_{Def,Det}$	Del_{Err}	B_d	B_i	B_r	B_{Err}	B_{Def}
GPS	2	+X	60.59	X	X	X	X	X	0	0	0	0	0
GPS	2	+X	72.35	X	X	X	X	X	0	0	0	0	0
GPS	2	+X	82.15	X	X	X	X	X	0	0	0	0	0
GPS	2	+X	87.47	X	X	X	X	X	0	0	0	0	0
GPS	2	+Y	60.59	X	X	X	X	X	0	0	0	0	0
GPS	2	+Y	72.35	X	X	X	X	X	0	0	0	0	0
GPS	2	+Y	82.15	X	X	X	X	X	0	0	0	0	0
GPS	2	+Y	87.47	X	X	X	X	X	0	0	0	0	0
GPS	2	-X	60.59	X	X	X	X	X	0	0	0	0	0
GPS	2	-X	72.35	X	X	X	X	X	0	0	0	0	0
GPS	2	-X	82.15	X	X	X	X	X	0	0	0	0	0
GPS	2	-X	87.47	X	X	X	X	X	0	0	0	0	0
GPS	2	-Y	60.59	77.67	17.08	X	X	X	1	1	0	0	0
GPS	2	-Y	72.35	77.67	5.32	X	X	X	1	1	0	0	0
GPS	2	-Y	82.15	X	X	X	X	X	0	0	0	0	0
GPS	2	-Y	87.47	X	X	X	X	X	0	0	0	0	0

Tableau A.1 – Injection de fautes sur le GPS : saut = 2 m

Comp	Saut	Dir	t_{Inj}	t_{Det}	Del_{Det}	Del_{Def}	$Del_{Def,Det}$	Del_{Err}	B_d	B_i	B_r	B_{Err}	B_{Def}
GPS	5	+X	60.59	65.35	4.76	X	X	3.92	1	1	0	1	0
GPS	5	+X	72.35	77.39	5.04	X	X	4.20	1	1	0	1	0
GPS	5	+X	82.15	X	X	X	X	0.00	0	0	0	1	0
GPS	5	+X	87.47	X	X	X	X	0.00	0	0	0	1	0
GPS	5	+Y	60.59	89.43	28.84	X	X	28.00	1	1	0	1	0
GPS	5	+Y	72.35	89.43	17.08	X	X	16.24	1	1	0	1	0
GPS	5	+Y	82.15	89.43	7.28	X	X	6.44	1	1	0	1	0
GPS	5	+Y	87.47	89.99	2.52	X	X	1.68	1	1	0	1	0
GPS	5	-X	60.59	89.99	29.40	X	X	28.56	1	1	0	1	0
GPS	5	-X	72.35	89.99	17.64	X	X	16.80	1	1	0	1	0
GPS	5	-X	82.15	89.99	7.84	X	X	7.00	1	1	0	1	0
GPS	5	-X	87.47	90.27	2.80	X	X	1.96	1	1	1	1	0
GPS	5	-Y	60.59	64.23	3.64	X	X	2.80	1	1	0	1	0
GPS	5	-Y	72.35	73.75	1.40	X	X	0.56	1	1	0	1	0
GPS	5	-Y	82.15	X	X	X	X	0.00	0	0	0	1	0
GPS	5	-Y	87.47	X	X	X	X	0.00	0	0	0	1	0

Tableau A.2 – Injection de fautes sur le GPS : : saut = 5 m

Comp	Saut	Dir	t_{Inj}	t_{Det}	Del_{Det}	Del_{Def}	$Del_{Def,Det}$	Del_{Err}	B_d	B_i	B_r	B_{Err}	B_{Def}
GPS	10	+X	60.59	62.55	1.96	X	X	1.68	1	1	1	1	0
GPS	10	+X	72.35	74.03	1.68	X	X	1.40	1	1	0	1	0
GPS	10	+X	82.15	82.71	0.56	X	X	0.28	1	1	1	1	0
GPS	10	+X	87.47	X	X	X	X	0.00	0	0	0	1	0
GPS	10	+Y	60.59	85.79	25.20	X	X	24.92	1	1	0	1	0
GPS	10	+Y	72.35	85.79	13.44	X	X	13.16	1	1	0	1	0
GPS	10	+Y	82.15	86.07	3.92	X	X	3.64	1	1	0	1	0
GPS	10	+Y	87.47	88.59	1.12	X	X	0.84	1	1	1	1	0
GPS	10	-X	60.59	70.39	9.80	X	X	9.52	1	1	0	1	0
GPS	10	-X	72.35	74.03	1.68	X	X	1.40	1	1	1	1	0
GPS	10	-X	82.15	87.19	5.04	X	X	4.76	1	1	0	1	0
GPS	10	-X	87.47	88.59	1.12	X	X	0.84	1	1	1	1	0
GPS	10	-Y	60.59	61.99	1.40	X	X	1.12	1	1	1	1	0
GPS	10	-Y	72.35	72.91	0.56	X	X	0.28	1	1	1	1	0
GPS	10	-Y	82.15	82.71	0.56	X	X	0.28	1	1	1	1	0
GPS	10	-Y	87.47	X	X	X	X	0.00	0	0	0	1	0

Tableau A.3 – Injection de fautes sur le GPS : saut = 10 m

Comp	Saut	Dir	t_{Inj}	t_{Det}	Del_{Det}	Del_{Def}	$Del_{Def,Det}$	Del_{Err}	B_d	B_i	B_r	B_{Err}	B_{Def}
GPS	20	+X	60.59	61.15	0.56	45.64	45.08	0.56	1	1	1	1	1
GPS	20	+X	72.35	72.91	0.56	45.08	44.52	0.56	1	1	1	1	1
GPS	20	+X	82.15	X	X	46.48	X	0.00	1	1	1	1	1
GPS	20	+X	87.47	88.31	0.84	X	X	0.84	1	1	1	1	0
GPS	20	+Y	60.59	61.71	1.12	46.48	45.36	1.12	1	1	1	1	1
GPS	20	+Y	72.35	74.59	2.24	46.20	43.96	2.24	1	1	1	1	1
GPS	20	+Y	82.15	83.27	1.12	45.36	44.24	1.12	1	1	1	1	1
GPS	20	+Y	87.47	87.75	0.28	X	X	0.28	1	1	1	1	0
GPS	20	-X	60.59	61.43	0.84	47.32	46.48	0.84	1	1	1	1	1
GPS	20	-X	72.35	72.91	0.56	44.80	44.24	0.56	1	1	1	1	1
GPS	20	-X	82.15	83.55	1.40	X	X	1.40	1	1	1	1	0
GPS	20	-X	87.47	88.03	0.56	X	X	0.56	1	1	1	1	0
GPS	20	-Y	60.59	61.15	0.56	45.36	44.80	0.56	1	1	1	1	1
GPS	20	-Y	72.35	X	X	45.08	X	0.00	1	1	1	1	1
GPS	20	-Y	82.15	82.43	0.28	45.08	44.80	0.28	1	1	1	1	1
GPS	20	-Y	87.47	88.87	1.40	X	X	1.40	1	1	1	1	0

Tableau A.4 – Injection de fautes sur le GPS : saut = 20 m

A.2 fautes injectées sur les encodeurs

Pour les encodeurs nous avons injecté 24 fautes, 4 fautes de type *trame bloquée*, 4 fautes de types *mise à zéro*, et 16 fautes de *biais* en simulant quatre biais différents (1, 2, 3, 4) que nous avons ajoutés aux sorties odometriques suivant quatre instants différents. Les tables (A.5,A.6, A.7, A.8, A.9 et A.10) montrent les résultats de ces scénarios de fautes.

Comp	Type	t_{Inj}	t_{Det}	Del_{Det}	Del_{Def}	$Del_{Def,Det}$	Del_{Err}	B_d	B_i	B_r	B_{Err}	B_{Def}
F-WSS	Hard	60.59	64.51	3.92	13.44	9.52	0.56	1	1	1	1	1
F-WSS	Hard	72.35	77.67	5.32	11.48	6.16	-0.28	1	1	1	1	1
F-WSS	Hard	82.15	87.47	5.32	8.12	2.80	2.24	1	1	1	1	1
F-WSS	Hard	87.47	99.79	12.32	13.72	1.40	5.04	1	1	1	1	1

Tableau A.5 – Injection de fautes de trame bloquée sur les encodeurs F-WSS

Comp	Type	t_{Inj}	t_{Det}	Del_{Det}	Del_{Def}	$Del_{Def,Det}$	Del_{Err}	B_d	B_i	B_r	B_{Err}	B_{Def}
F-WSS	Null	60.59	62.27	1.68	2.52	0.84	1.12	1	1	1	1	1
F-WSS	Null	72.35	73.47	1.12	4.20	3.08	0.28	1	1	1	1	1
F-WSS	Null	82.15	82.71	0.56	2.52	1.96	0.00	1	1	1	1	1
F-WSS	Null	87.47	90.83	3.36	X	X	2.52	1	1	0	1	0

Tableau A.6 – Injection de fautes de mise à zéro sur les encodeurs F-WSS

Comp	Bias	t_{Inj}	t_{Det}	Del_{Det}	Del_{Def}	$Del_{Def,Det}$	Del_{Err}	B_d	B_i	B_r	B_{Err}	B_{Def}
F-WSS	1	60.59	64.51	3.92	17.08	13.16	1.12	1	1	1	1	1
F-WSS	1	72.35	85.23	12.88	33.88	21.00	10.08	1	1	0	1	1
F-WSS	1	82.15	94.19	12.04	21.56	9.52	9.24	1	1	0	1	1
F-WSS	1	87.47	94.19	6.72	24.36	17.64	5.88	1	1	0	1	1

Tableau A.7 – Injection de fautes de biais de 1 m/s sur les encodeurs F-WSS

Comp	Bias	t_{Inj}	t_{Det}	Del_{Det}	Del_{Def}	$Del_{Def,Det}$	Del_{Err}	B_d	B_i	B_r	B_{Err}	B_{Def}
F-WSS	2	60.59	63.39	2.80	14.56	11.76	1.40	1	1	1	1	1
F-WSS	2	72.35	77.67	5.32	12.88	7.56	3.64	1	1	1	1	1
F-WSS	2	82.15	87.19	5.04	14.84	9.80	3.64	1	1	1	1	1
F-WSS	2	87.47	95.59	8.12	18.48	10.36	6.72	1	1	0	1	1

Tableau A.8 – Injection de fautes de biais de 2 m/s sur les encodeurs F-WSS

Comp	Bias	t_{Inj}	t_{Det}	Del_{Det}	Del_{Def}	$Del_{Def,Det}$	Del_{Err}	B_d	B_i	B_r	B_{Err}	B_{Def}
F-WSS	3	60.59	62.83	2.24	13.16	10.92	1.12	1	1	1	1	1
F-WSS	3	72.35	76.27	3.92	6.44	2.52	2.80	1	1	1	1	1
F-WSS	3	82.15	86.07	3.92	15.96	12.04	2.80	1	1	1	1	1
F-WSS	3	87.47	91.11	3.64	11.20	7.56	0.84	1	1	1	1	1

Tableau A.9 – Injection de fautes de biais de 3 m/s sur les encodeurs F-WSS

Comp	Bias	t_{Inj}	t_{Det}	Del_{Det}	Del_{Def}	$Del_{Def,Det}$	Del_{Err}	B_d	B_i	B_r	B_{Err}	B_{Def}
F-WSS	4	60.59	62.55	1.96	5.32	3.36	1.12	1	1	1	1	1
F-WSS	4	72.35	75.71	3.36	4.76	1.40	2.52	1	1	1	1	1
F-WSS	4	82.15	85.23	3.08	7.28	4.20	2.24	1	1	1	1	1
F-WSS	4	87.47	89.15	1.68	8.12	6.44	0.84	1	1	1	1	1

Tableau A.10 – Injection de fautes de biais de 4 m/s sur les encodeurs F-WSS

A.3 Fautes injectées sur l'angle de braquage

Nous avons simulé deux fautes significatives sur l'angle de braquage, ce type de faute bloque le capteur au moment où le véhicule tourne. En effet à cet instant le capteur est censé délivrer une sortie différente de zéro. On voit bien qu'une telle faute est bien détectée, diagnostiquée et recouverte par notre architecture comme indiqué dans la table A.11.

Comp	Type	t_{Inj}	t_{Det}	Del_{Det}	Del_{Def}	$Del_{Def,Det}$	Del_{Err}	B_d	B_i	B_r	B_{Err}	B_{Def}
Angle de braquage	Null	59.75	67.31	7.56	9.52	1.96	1.40	1	1	1	1	1
Angle de braquage	Null	82.15	88.03	5.88	13.16	7.28	1.12	1	1	1	1	1

Tableau A.11 – Injection de fautes sur le capteur d'angle de braquage

A.4 Fautes logicielles injectées dans les filtres de Kalman

La table A.12 présente les effets des différentes fautes logicielles injectées sur les modèles du système et d'observation des deux filtres de Kalman. Ces injections sont faites par mutations des éléments des matrices de transition A, de commande B, et d'observation H en modifiant directement le code exécuté par les filtres de Kalman.

Original	Mutant	t_{Inj}	t_{Det}	Del_{Det}	Del_{Def}	$Del_{Def,Det}$	Del_{Err}	B_d	B_i	B_r	B_{Err}	B_{Def}
Q1	Q1 = 0.00001.R1	0.00	87.19	87.19	X	X	25.76	1	0	0	1	0
$A^1_{1,4} : cos(\theta)$	$A^1_{1,4} : sin(\theta)$	0.00	72.63	72.63	X	X	19.32	1	0	0	1	0
$A^1_{2,4} : sin(\theta)$	$A^1_{2,4} : cos(\theta)$	0.00	71.23	71.23	X	X	18.20	1	0	0	1	0
$A^1_{1,4} : cos(\theta)$	$A^1_{1,4} : -cos(\theta)$	0.00	72.07	72.07	X	X	18.76	1	0	0	1	0
$A^1_{2,4} : sin(\theta)$	$A^1_{2,4} : -sin(\theta)$	0.00	72.07	72.07	X	X	19.04	1	0	0	1	0
$B^1_{1,1} : \frac{1}{2}dt^2$	$B^1_{1,1} : dt^2$	0.00	X	X	X	X	X	0	0	0	0	0
$B^1_{1,1} : \frac{1}{2}dt^2$	$B^1_{1,1} : \frac{1}{4}dt^2$	0.00	X	X	X	X	X	0	0	0	0	0
$H^1_{1,1} : 1$	$H^1_{1,1} : 0$	0.00	88.03	88.03	89.99	1.96	21.28	1	0	0	1	1
$H^1_{1,2} : 0$	$H^1_{1,2} : 1$	0.00	56.11	56.11	58.07	1.96	2.52	1	0	0	1	1
$H^1_{1,1} : 1$	$H^1_{1,1} : -1$	0.00	53.87	53.87	55.83	1.96	1.40	1	0	0	1	1
Q2	Q2 = 0.00001R2	0.00	84.67	84.67	X	X	24.92	1	0	0	1	0
$A^2_{1,1} : 1$	$A^2_{1,1} : 0$	0.00	55.83	55.83	57.79	1.96	3.08	1	0	0	1	1
$A^2_{1,1} : 1$	$A^2_{1,1} : -1$	0.00	55.83	55.83	57.79	1.96	3.08	1	0	0	1	1
$A^2_{1,2} : 0$	$A^2_{1,2} : 1$	0.00	55.83	55.83	57.79	1.96	3.36	1	0	0	1	1
$B^2_{1,1} : cos\theta$	$B^2_{1,1} : sin\theta$	0.00	57.79	57.79	X	X	5.60	1	0	0	1	0
$B^2_{2,1} : sin\theta$	$B^2_{2,1} : cos\theta$	0.00	57.79	57.79	X	X	5.32	1	0	0	1	0
$B^2_{1,1} : dt$	$B^2_{1,1} : dt/2$	0.00	X	X	X	X	5.32	0	0	0	1	0
$H^2_{1,1} : 1$	$H^2_{1,1} : 0$	0.00	X	X	X	X	5.32	0	0	0	1	0
$H^2_{1,2} : 0$	$H^2_{1,2} : 1$	0.00	55.55	55.55	58.35	2.80	2.52	1	0	0	1	1
$H^2_{1,1} : 1$	$H^2_{1,1} : -1$	0.00	54.71	54.71	55.83	1.12	2.24	1	0	0	1	1

Tableau A.12 – Injection de fautes logicielles par mutation

A.5 Mesures globales sur toute l'expérimentation

La table A.13 présente les différents taux de diagnostic, de rétablissement, de faux positifs et d'absence de détection de toute notre campagne d'expérimentation

Type de faute	P_i (%)	P_r (%)	P_{FP} (%)	P_{ND} (%)	P_{ESD}(%)
Matérielle	100.00	62.32	0.00	0.00	44
Logicielle			0.00	0.00	44

Tableau A.13 – Mesures sur les fautes matérielles et logicielles

Bibliographie

[cen, 2011] (2011). Cenelec 50128 : Railway applications-communication, signalling and processing systems-software for railway control and protection systems.

[Abuhadrous, 2005] Abuhadrous, I. (2005). *Système embarqué temps réel de localisation et de modélisation 3D par fusion multi-capteur.* PhD thesis, École Nationale Supérieure des Mines de Paris.

[Allerton and Jia, 2008] Allerton, D. and Jia, H. (2008). Distributed data fusion algorithms for inertial network systems. *Radar, Sonar & Navigation, IET*, 2(1) :51–62.

[Appriou, 1991] Appriou, A. (1991). Probabilités et incertitude en fusion de données multi-senseurs. *Tiré à part- Office national d'études et de recherches aerospatiales.*

[Appriou, 2002] Appriou, A. (2002). Discrimination multisignal par la théorie de l'évidence. *Décision et Reconnaissance des formes en signal, Hermes Science Publication*, pages 219–258.

[Aquirre-Martinez, 2013] Aquirre-Martinez, F. (2013). *Prise en compte des incertitudes dans les méthodes d'analyse de la sûreté de fonctionnement : application à l'intégration du facteur humain dans les analyses des risques appliqués aux transports.* PhD thesis, Université de Technologie de Compiegne UTC.

[Arlat et al., 1993] Arlat, J., Costes, A., Crouzet, Y., Laprie, J.-C., and Powell, D. (1993). Fault injection and dependability evaluation of fault-tolerant systems. *IEEE Transactions on Computers*, 42(8) :913–923.

[Avizienis and Chen, 1977] Avizienis, A. and Chen, L. (1977). On the implementation of n-version programming for software fault tolerance during execution. In *Proceeding of the First IEEE-CS International Computer Software and Applications Conference (COM PSAC 77), Chicago.*

[Avizienis et al., 2004] Avizienis, A., Laprie, J., Randell, B., and Landwehr, C. (2004). Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable Secure Computing*, 1(1) :11–33.

[Bloch, 2003] Bloch, I. (2003). *Fusion d'informations en traitement du signal et des images*. Hermes Science Publications.

[Bloch and Maître, 1997] Bloch, I. and Maître, H. (1997). Data fusion in 2d and 3d image processing : an overview. In *Proceedings of the 10th Brazilian Symposium on Computer Graphics and Image Processing*, pages 127–134.

[Borenstein and Feng, 1996] Borenstein, J. and Feng, L. (1996). Measurement and correction of systematic odometry errors in mobile robots. *IEEE Transactions on Robotics and Automation*, 12(6) :869–880.

[Boström et al., 2007] Boström, H., Andler, S. F., Brohede, M., Johansson, R., Karlsson, A., Van Laere, J., Niklasson, L., Nilsson, M., Persson, A., and Ziemke, T. (2007). On the definition of information fusion as a field of research. Technical Report HS-IKI-TR-07-006, School of humanities and Informatics, University of Skövde.

[Bradai, 2007] Bradai, B. (2007). *Optimisation des Lois de Commande d'Éclairage Automobile par Fusion de Données*. PhD thesis, Mulhouse.

[Casanova et al., 2008] Casanova, O. L. et al. (2008). Robot position tracking using kalman filter. In *Proceedings of the World Congress on Engineering,London UK*, volume II.

[Caspi and Salem, 2000] Caspi, P. and Salem, R. (2000). Threshold and bounded-delay voting in critical control systems. In *Formal Techniques in Real-Time and Fault-Tolerant Systems*, pages 327–337. Springer.

[Chebbah et al., 2011] Chebbah, M., Martin, A., Yaghlane, B. B., et al. (2011). Estimation de la fiabilité des sources des bases de données évidentielles. *Revue Nationale des Technologies de l'Information*, (21) :191–208.

[Daran, 1996] Daran, M. (1996). *Modélisation des comportements érronés du logiciel et application à la validation des tests par injection de fautes*. PhD thesis, Institut National Polytechnique de Toulouse-INPT.

[De Luca et al., 1998] De Luca, A., Oriolo, G., and Samson, C. (1998). Feedback control of a nonholonomic car-like robot. In *Robot motion planning and control*, pages 171–253. Springer.

[Delmotte, 2007] Delmotte, F. (2007). Detection of defective sources in the setting of possibility theory. *Fuzzy sets and systems*, 158(5) :555–571.

[Delmotte and Gacquer, 2008] Delmotte, F. and Gacquer, G. (2008). Detection of defective sources with belief functions. In *Proceedings of IPMU, Malaga, Spain*, pages 337–344.

[Démotier et al., 2006] Démotier, S., Schon, W., and Denoeux, T. (2006). Risk assessment based on weak information using belief functions : a case study in water treatment. *IEEE Transactions on Systems, Man, and Cybernetics, Part C : Applications and Reviews,*, 36(3) :382–396.

[Dempster, 1967] Dempster, A. P. (1967). Upper and lower probabilities induced by a multivalued mapping. *The annals of mathematical statistics*, 38(2) :325–339.

[Denoeux, 1995] Denoeux, T. (1995). A k-nearest neighbor classification rule based on dempster-shafer theory. *IEEE Transactions on Systems, Man and Cybernetics*, 25(5) :804–813.

[Dubois and Prade, 1988a] Dubois, D. and Prade, H. (1988a). *Possibility theory*. Plenum Press, New-York.

[Dubois and Prade, 1988b] Dubois, D. and Prade, H. (1988b). Representation and combination of uncertainty with belief functions and possibility measures. *Computational Intelligence*, 4(3) :244–264.

[Dubois and Prade, 1994] Dubois, D. and Prade, H. (1994). Possibility theory and data fusion in poorly informed environments. *Control Engineering Practice*, 2(5) :811–823.

[Dubois and Prade, 2000] Dubois, D. and Prade, H. (2000). Possibility theory in information fusion. In *Proceedings of the Third International Conference on Information Fusion*, volume 1, pages PS6–P19.

[Durrant-Whyte, 1988] Durrant-Whyte, H. F. (1988). Sensor models and multisensor integration. *The International Journal of Robotics Research*, 7(6) :97–113.

[Escamilla-Ambrosio and Mort, 2001] Escamilla-Ambrosio, P. and Mort, N. (2001). A hybrid kalman filter-fuzzy logic multisensor data fusion architecture with fault tolerant characteristics. In *proceedings of the international conference on artificial intelligence*, pages 361–367.

[Freeston, 2002] Freeston, L. (2002). Applications of the kalman filter algorithm to robot localisation and world modelling. *Electrical Engineering Final Year Project. University of Newcastle, Australia.*

[Frémont, 2009] Frémont, V. (2009). Odométrie 3d vision/lidar pour les véhicules intelligents. In *Journées Nationales de la Recherche en Robotique, JNRR.*

[Grandin, 2006] Grandin, J.-F. (2006). Fusion de données - theories et méthodes, thechniques d'ingenieurs.

[Hall and Llinas, 1997] Hall, D. and Llinas, J. (1997). An introduction to multisensor data fusion. *Proceeding of the IEEE*, 85(1) :6–23.

[Hall and McMullen, 2004] Hall, D. and McMullen, S. (2004). *Mathematical techniques in multisensor data fusion.* Artech House Publishers.

[Hu et al., 2003] Hu, C., Chen, W., Chen, Y., and Liu, D. (2003). Adaptive kalman filtering for vehicle navigation. *Journal of Global Positioning Systems*, 2(1) :42–47.

[Jarbaoui, 2003] Jarbaoui, M. T. (2003). *Sûreté de fonctionnement de systèmes informatiques : étalonnage et représentativité des fautes.* PhD thesis, Toulouse, INPT.

[Kalman and Bucy, 1961] Kalman, R. E. and Bucy, R. S. (1961). New results in linear filtering and prediction theory. *Journal of Basic Engineering*, 83(3) :95–108.

[Kalman et al., 1960] Kalman, R. E. et al. (1960). A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1) :35–45.

[Khaleghi et al., 2009] Khaleghi, B., Razavi, S. N., Khamis, A., Karray, F. O., and Kamel, M. (2009). Multisensor data fusion : Antecedents and directions. In *3rd International Conference on Signals, Circuits and Systems (SCS)*, pages 1–6.

[Kim et al., 2007] Kim, S.-G., Crassidis, J. L., Cheng, Y., Fosbury, A. M., and Junkins, J. L. (2007). Kalman filtering for relative spacecraft attitude and position estimation. *Journal of Guidance, Control, and Dynamics*, 30(1) :133–143.

[Lancaster and Seneta, 1969] Lancaster, H. O. and Seneta, E. (1969). *Chi-Square Distribution*. Wiley Online Library.

[Laprie et al., 1992] Laprie, J., Avizienis, A., and Kopetz, H. (1992). *Dependability : Basic Concepts and Terminology*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

[Laprie et al., 1990] Laprie, J.-C., Arlat, J., Beounes, C., and Kanoun, K. (1990). Definition and analysis of hardware-and software-fault-tolerant architectures. *Computer*, 23(7) :39–51.

[Lefevre et al., 2002] Lefevre, E., Colot, O., and Vannoorenberghe, P. (2002). Belief function combination and conflict management. *Information Fusion*, 3(2) :149–162.

[Lefevre et al., 2000] Lefevre, E., Colot, O., Vannoorenberghe, P., and De Brucq, D. (2000). A generic framework for resolving the conflict in the combination of belief structures. In *Proceedings of the 3rd International Conference on Information Fusion.*, volume 1, pages MOD4 11–18.

[Lohweg et al., 2011] Lohweg, V., Voth, K., and Glock, S. (2011). A possibilistic framework for sensor fusion with monitoring of sensor reliability. *Sensor Fusion-Foundation and Applications, intechopen. com*, pages 191–226.

[Lussier, 2007] Lussier, B. (2007). *Tolérance aux fautes dans les systèmes autonomes*. PhD thesis, Institut National Polytechnique de Toulouse.

[Martin, 2005] Martin, A. (2005). La fusion d'informations. *Polycopié de cours ENSIETA-Réf*, 1484 :117.

[Masson, 2005] Masson, M.-H. (2005). *Apports de la théorie des possibilités et des fonctions de croyance à l ?analyse de données imprécises*. PhD thesis, Université de Technologie de Compiègne.

[Mitchell, 2007] Mitchell, H. (2007). *Multi-sensor data fusion : an introduction*. Springer.

[Morales et al., 2008] Morales, Y., Takeuchi, E., and Tsubouchi, T. (2008). Vehicle localization in outdoor woodland environments with sensor fault detection. In *International Conference on Robotics and Automation (ICRA)*, pages 449–454.

[Moras, 2013] Moras, J. (2013). *Grilles de perception évidentielles pour la navigation robotique en milieu urbain*. PhD thesis, Université de Technologie de Compiègne.

[Okello, 1993] Okello, N.N., T. S. (1993). Design of data fusion system for multiradar target detection. In *International Conference on Digital Object Identifier : 10.1109/ICSMC.1993.385094*, volume 3, pages 672–677.

[Peng et al., 2012] Peng, M., He, J., Shen, M., and Xie, K. (2012). Analog fault diagnosis using decision fusion. In *7th International Conference on Computer Science & Education (ICCSE)*, pages 17–20.

[Randell, 1975] Randell, B. (1975). System structure for software fault tolerance. *IEEE Transaction on Software Engennering*, 1(2).

[Ricquebourg et al., 2007a] Ricquebourg, V., Delafosse, M., Delahoche, L., Marhic, B., Jolly, A., and Menga, D. (2007a). Combinaison de sources de données pour la détection de dysfonctionnement capteur. LFA.

[Ricquebourg et al., 2007b] Ricquebourg, V., Delafosse, M., Delahoche, L., Marhic, B., Jolly-Desodt, A., and Menga, D. (2007b). Fault detection by combining redundant sensors : a conflict approach within the tbm framework. *Cognitive Systems with Interactive Sensors, COGIS*.

[Rouchouze, 1994] Rouchouze, C. (1994). Fusion de données : exemples défense et axes de recherche. *TS. Traitement du signal*, 11(6) :459–464.

[Roumeliotis et al., 1998] Roumeliotis, S. I., Sukhatme, G. S., and Bekey, G. A. (1998). Sensor fault detection and identification in a mobile robot. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, pages 1383–1388.

[Shafer, 1976] Shafer, G. (1976). *A mathematical theory of evidence*, volume 1. Princeton university press Princeton.

[Shu-qing and Sheng-xiu, 2010] Shu-qing, L. and Sheng-xiu, Z. (2010). A congeneric multi-sensor data fusion algorithm and its fault-tolerance. In *International*

Conference on Computer Application and System Modeling (ICCASM), volume 1, pages V1–339.

[Smets, 1990] Smets, P. (1990). The combination of evidence in the transferable belief model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(5) :447–458.

[Smets, 1993] Smets, P. (1993). Belief functions : the disjunctive rule of combination and the generalized bayesian theorem. *International Journal of approximate reasoning*, 9(1) :1–35.

[Smets, 1998] Smets, P. (1998). The transferable belief model for quantified belief representation. *Handbook of defeasible reasoning and uncertainty management systems*, 1 :267–301.

[Smets, 2000] Smets, P. (2000). Data fusion in the transferable belief model. In *Proceedings of the Third International Conference on Information Fusion*, volume 1, pages PS21–PS33.

[Smets, 2005] Smets, P. (2005). Decision making in the tbm : the necessity of the pignistic transformation. *International Journal of Approximate Reasoning*, 38(2) :133–147.

[Smets and Kennes, 1994] Smets, P. and Kennes, R. (1994). The transferable belief model. *Artificial intelligence*, 66(2) :191–234.

[Uhlmann, 2003] Uhlmann, J. K. (2003). Covariance consistency methods for fault-tolerant distributed data fusion. *Information Fusion*, 4(3) :201–215.

[Vannoorenberghe, 2003] Vannoorenberghe, P. (2003). Un état de l ?art sur les fonctions de croyance appliquées au traitement de l ?information. *Revue I3*, 3(1) :9–45.

[Vannoorenberghe and Denoeux, 2002] Vannoorenberghe, P. and Denoeux, T. (2002). Handling uncertain labels in multiclass problems using belief decision trees. In *Proceedings of IPMU*, volume 3, pages 1919–1926.

[White, 1991] White, F. E. (1991). Data fusion lexicon. Technical report, DTIC Document.

[Yager, 1987] Yager, R. R. (1987). On the dempster-shafer framework and new combination rules. *Information sciences*, 41(2) :93–137.

[Yi et al., 2000] Yi, Z., Khing, H., Seng, C., and Wei, Z. (2000). Multi-ultrasonic sensor fusion for mobile robots. In *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*, pages 387–391.

[Zhuo-hua et al., 2005] Zhuo-hua, D., Zi-xing, C., and Jin-xia, Y. (2005). Fault diagnosis and fault tolerant control for wheeled mobile robots under unknown environments : A survey. In *Proceedings of the International Conference on Robotics and Automation, ICRA*, pages 3428–3433.

[Zug and Kaiser, 2009] Zug, S. and Kaiser, J. (2009). An approach towards smart fault-tolerant sensors. In *Proceedings of the International Workshop on Robotic and Sensors Environments. ROSE*, pages 35–40.